



• • • •

[illegible]

```
0001 0 MODULE DBGENCDEC (IDENT = 'V04-000') =
0002 1 BEGIN
0003 1
0004 1 *****
0005 1 *
0006 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0007 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0008 1 *   ALL RIGHTS RESERVED.
0009 1 *
0010 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0011 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0012 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0013 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0014 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0015 1 *   TRANSFERRED.
0016 1 *
0017 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0018 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0019 1 *   CORPORATION.
0020 1 *
0021 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0022 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0023 1 *
0024 1 *****
0025 1 *****
0026 1
0027 1 ++
0028 1
0029 1   Original Author: John Francis
0030 1
0031 1   Modification history:
0032 1
0033 1   001   Walter Carrell III, 3-Jun-83
0034 1   ASHP thought it had only 5 arguments. See the comment for
0035 1   the MACRO Opcode_list for a more complete explanation.
0036 1
0037 1   002   Walter Carrell III, 08-Jun-83
0038 1   The outside world expects DBG$OPCODE_INDEX to return an index
0039 1   into DBG$Opcode_Kind_Table. Within DBGENCDEC DBG$OPCODE_INDEX
0040 1   was expected to return an index into DBG$Opcode_Name_Table.
0041 1   DBG$OPCODE_INDEX originally returned an index into
0042 1   DBG$Opcode_Name Table. This edit changes that. The place
0043 1   where DBG$OPCODE_INDEX was called within DBGENCDEC have been
0044 1   changed to call Opcode Name Index, a local routine which
0045 1   is the original DBG$OPCODE_INDEX with some fixed to make
0046 1   it work correctly. A new DBG$OPCODE_INDEX was written to
0047 1   return an index into DBG$Opcode_Kind_table.
0048 1
0049 1   003   Walter Carrell III, 09-Jun-83
0050 1   1. CVTTP was out of order in the Mnemonic table.
0051 1   2. Allow ^Y out of the printing of destinations of a CASE list.
0052 1   3. The limit of CASE statements was always being read as a LONG
0053 1   instead of in the appropriate type.
0054 1   4. XFC and BUGx had a bad table entries.
0055 1
0056 1   004   Walter Carrell III, 13-Jun-83
0057 1   All the 2 byte opcode instructions had FE instead of FD in the
```



```
58      0058 1      table for the first byte of the opcode.
59      0059 1
60      0060 1      005  Walter Carrell III, 20-Jun-83
61      0061 1      1. ^S and ^I were not allowed
62      0062 1      2. Addr-exprs were not allowed in any context other than
63      0063 1      branch displacements
64      0064 1      3. (Rn)[Rn] where Rn is the same for both was not caught
65      0065 1      4. Choose displacement sizes instead of defaulting to Byte.
66      0066 1      5. Allow SP in indexed mode, [SP]
67      0067 1      6. Correct the CASE offset address calculation
68      0068 1      7. Correct the context of ASHP
69      0069 1      8. Make DEP/INSTR for I^# work
70      0070 1
71      0071 1      006  Walter Carrell III, 11-Jul-83
72      0072 1      Fix Parse_Register so it doesn't use Parse_Expression
73      0073 1      Parse_Expression cannot be used because a scope may
74      0074 1      not be active the a DEP/INSTR is issued and therefore
75      0075 1      Rn is thought to be a symbol outside the active scope.
76      0076 1
77      0077 1      007  Walter Carrell III, 08-Aug-83
78      0078 1      Enhance error reporting of instruction Encoding
79      0079 1
80      0080 1      008  Walter Carrell III, 20-Sep-83
81      0081 1      Fix instruction encoding to allow quad and octaword literals
82      0082 1
83      0083 1      --
84      0084 1
85      0085 1      REQUIRE 'SRC$:DBGPROLOG.REQ';
86      0219 1
87      0220 1      FORWARD ROUTINE
88      0221 1
89      0222 1      Global Routines
90      0223 1
91      0224 1      DBG$Ins_Decode,      ! Decode Single Instruction
92      0225 1      DBG$Ins_Encode,      ! Encode Single instruction
93      0226 1      DBG$OpCode_Index,    ! Convert Mnemonic to kind table index      ! Changed to return Kind tabl
94      0227 1                                     ! instead of name table index.
95      0228 1
96      0229 1      Local Routines
97      0230 1
98      0231 1      Opcode_Name_Index,    ! Convert Mnemonic to name table index      !
99      0232 1      Fetch_Instruction      ! Fetch bytes from instruction stream
100     0233 1      Fetch_Operand          ! Fetch (and optionally print) an operand
101     0234 1      Print_Address          ! Print operand address
102     0235 1      Print_Operand          ! Print operand value
103     0236 1      Parse_Operand          ! Parse one instruction operand
104     0237 1      Parse_Expression,      ! Parse an operand (address or value)
105     0238 1      Parse_Register,        ! Parse a register name
106     0239 1      Check_Register,        ! See if address describes a register
107     0240 1      Store_Operand          ! Store bytes in output stream
108     0241 1      Scan_Operand,          ! Separate one operand string
109     0242 1      Skip_Leading_Blanks : NOVALUE; ! Skip over leading spaces and/or tabs
110     0243 1
111     0244 1      EXTERNAL
112     0245 1      DBG$GB_RADIX: VECTOR[3, BYTE];      ! Radix settings
113     0246 1
114     0247 1
```



```
115 0248 1 EXTERNAL ROUTINE
116 0249 1   DBG$CONV TEXT VALUE,
117 0250 1   DBG$COVER_DX_DX,
118 0251 1   DBG$Print : NOVALUE,
119 0252 1   DBG$Print_Value : NOVALUE,
120 0253 1   DBG$Print_Identifier_PC : NOVALUE,
121 0254 1   DBG$NewLine : NOVALUE,
122 0255 1   DBG$Pop_Tempmem : NOVALUE,
123 0256 1   DBG$Push_Tempmem,
124 0257 1   DBG$Is_IE_Entry,
125 0258 1   DBG$Make_Val_Desc,
126 0259 1   DBG$Nparse_Address,
127 0260 1   DBG$Nparse_Expression,
128 0261 1   DBG$Prim_to_Val;
129 0262 1
130 0263 1 LITERAL
131 0264 1   simple_0_operand = ZX'00',
132 0265 1   simple_1_operand = ZX'01',
133 0266 1   simple_2_operand = ZX'02',
134 0267 1   simple_3_operand = ZX'03',
135 0268 1   branch_0_operand = ZX'04',
136 0269 1   branch_1_operand = ZX'05',
137 0270 1   branch_2_operand = ZX'06',
138 0271 1   branch_3_operand = ZX'07',
139 0272 1   convert_datatype = ZX'08',
140 0273 1   evaluate_address = ZX'08',
141 0274 1   simple_bit_field = ZX'08',
142 0275 1   routine_dispatch = ZX'08',
143 0276 1   locate_character = ZX'08',
144 0277 1   polynomial_value = ZX'08',
145 0278 1   probe_for_access = ZX'08',
146 0279 1   trailing_operand = ZX'09',
147 0280 1   string_3_operand = ZX'0A',
148 0281 1   string_4_operand = ZX'0B',
149 0282 1   string_5_operand = ZX'0C',
150 0283 1   string_6_operand = ZX'0D',
151 0284 1
152 0285 1   complex_SHIFT = ZX'10',
153 0286 1   complex_CASE = ZX'11',
154 0287 1   complex_EDIV = ZX'12',
155 0288 1   complex_EMOD = ZX'13',
156 0289 1   complex_EMUL = ZX'14',
157 0290 1   complex_INDEX = ZX'15',
158 0291 1   complex_CRC = ZX'16',
159 0292 1   complex_ASHP = ZX'17',
160 0293 1
161 0294 1   maximum_state = ZX'17',
162 0295 1
163 0296 1   context_b = ZX'00',
164 0297 1   context_w = ZX'01',
165 0298 1   context_l = ZX'02',
166 0299 1   context_q = ZX'03',
167 0300 1   context_o = ZX'04',
168 0301 1   context_f = ZX'05',
169 0302 1   context_d = ZX'06',
170 0303 1   context_g = ZX'07',
171 0304 1   context_h = ZX'08',
```

DBGENCDEC  
V04-000

L 2  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 4  
(1)

:	172	0305	1	context_bu	=	XX'09':				
:	173	0306	1	context_wu	=	XX'0A':				
:	174	0307	1	context_t	=	XX'0B':				
:	175	0308	1	context_p	=	XX'0C':	:	size.wu	base.b	
:	176	0309	1	context_m	=	XX'0D':	:	size.wu	base.b	
:	177	0310	1	context_v	=	XX'0E':	:	pos.l	size.b	base.b
:							:	pos.l	base.b	

```
179 0311 1 ++
180 0312 1
181 0313 1
182 0314 1
183 0315 1
184 0316 1
185 0317 1
186 0318 1
187 0319 1
188 0320 1
189 0321 1
190 0322 1
191 0323 1
192 0324 1
193 0325 1
194 0326 1
195 0327 1
196 0328 1
197 0329 1
198 0330 1
199 0331 1
200 0332 1
201 0333 1
202 0334 1
203 0335 1
204 0336 1
205 0337 1
206 0338 1
207 0339 1
208 0340 1
209 0341 1
210 0342 1
211 0343 1
212 0344 1
213 0345 1
214 0346 1
215 0347 1
216 0348 1
217 0349 1
218 M 0350 1
219 M 0351 1
220 M 0352 1
221 M 0353 1
222 M 0354 1
223 M 0355 1
224 M 0356 1
225 M 0357 1
226 M 0358 1
227 M 0359 1
228 M 0360 1
229 M 0361 1
230 M 0362 1
231 M 0363 1
232 M 0364 1
233 M 0365 1
234 M 0366 1
235 M 0367 1
```

The following table is used to build 2 data structures:

DBG\$Opcode\_Name\_Table - An alphabetical table of the Opcode names  
DBG\$Opcode\_Kind\_Table - A back translation table to get from an  
Op code to the Name.

Opcode\_entry is a macro the is defined twice to pass over  
Opcode\_List twice to build the two tables.

The arguments have the following definition:

1. The first argument is a flag that indicates that the entry has a duplicate Opcode and that it should be ignored in DBG\$Opcode\_Kind\_Table.
2. The second argument is the MNEMONIC. It must be 6 characters.
3. The third argument is the Opcode.
4. The fourth argument is the state to start with in the finite state machine
5. The fifth and optional sixth arguments are context flags. Their nature is not fully understood.

Note that the table must be in alphabetical order by MNEMONIC.  
A binary search is used to find table entries.

The editorial starts here.

The table is more complex than necessary for the simple matter of decoding for output and encoding instructions for deposit. The number of arguments would have been sufficient, instead of the last 3 arguments. The first byte of an argument tells you what you need to know about the rest of the argument.

The intent was apparently to have enough information in the table to allow the decoding of the instructions for interpreting watch points in the stack and registers. The information in the table is not sufficient for that purpose.

MACRO Opcode\_List =

```
Opcode_Entry(1,'ACBB',ZX'9D',branch_3_operand,context_b,context_w),
Opcode_Entry(1,'ACBD',ZX'6F',branch_3_operand,context_d,context_w),
Opcode_Entry(1,'ACBF',ZX'4F',branch_3_operand,context_f,context_w),
Opcode_Entry(1,'ACBG',ZX'4FFD',branch_3_operand,context_g,context_w),
Opcode_Entry(1,'ACBH',ZX'6FFD',branch_3_operand,context_h,context_w),
Opcode_Entry(1,'ACBL',ZX'F1',branch_3_operand,context_l,context_w),
Opcode_Entry(1,'ACBW',ZX'3D',branch_3_operand,context_w,context_w),
Opcode_Entry(1,'ADAW1',ZX'58',simple_2_operand,context_w),
Opcode_Entry(1,'ADDB2',ZX'80',simple_2_operand,context_b),
Opcode_Entry(1,'ADDB3',ZX'81',simple_3_operand,context_b),
Opcode_Entry(1,'ADDD2',ZX'60',simple_2_operand,context_d),
Opcode_Entry(1,'ADDD3',ZX'61',simple_3_operand,context_d),
Opcode_Entry(1,'ADDF2',ZX'40',simple_2_operand,context_f),
Opcode_Entry(1,'ADDF3',ZX'41',simple_3_operand,context_f),
Opcode_Entry(1,'ADDG2',ZX'40FD',simple_2_operand,context_g),
Opcode_Entry(1,'ADDG3',ZX'41FD',simple_3_operand,context_g),
Opcode_Entry(1,'ADDH2',ZX'60FD',simple_2_operand,context_h),
```



236	M	0368	1	Opcode_Entry(1, 'ADDH3', 'ZX'61FD', ,simple_3_operand,context_h),	!
237	M	0369	1	Opcode_Entry(1, 'ADDL2', 'ZX'C0', ,simple_2_operand,context_l),	
238	M	0370	1	Opcode_Entry(1, 'ADDL3', 'ZX'C1', ,simple_3_operand,context_l),	
239	M	0371	1	Opcode_Entry(1, 'ADDP4', 'ZX'20', ,simple_2_operand,context_p),	
240	M	0372	1	Opcode_Entry(1, 'ADDP6', 'ZX'21', ,simple_3_operand,context_p),	
241	M	0373	1	Opcode_Entry(1, 'ADDW2', 'ZX'A0', ,simple_2_operand,context_w),	
242	M	0374	1	Opcode_Entry(1, 'ADDW3', 'ZX'A1', ,simple_3_operand,context_w),	
243	M	0375	1	Opcode_Entry(1, 'ADWC', 'ZX'D8', ,simple_2_operand,context_l),	
244	M	0376	1	Opcode_Entry(1, 'AOBLEQ', 'ZX'F3', ,branch_2_operand,context_l,context_b),	
245	M	0377	1	Opcode_Entry(1, 'AOBLSS', 'ZX'F2', ,branch_2_operand,context_l,context_b),	
246	M	0378	1	Opcode_Entry(1, 'ASHL', 'ZX'78', ,complex_SHIFT,context_l),	
247	M	0379	1	Opcode_Entry(1, 'ASHP', 'ZX'F8', ,complex_ASHP,context_b,context_p),	! simple_3_operand to comple
248	M	0380	1	Opcode_Entry(1, 'ASHQ', 'ZX'79', ,complex_SHIFT,context_q),	
249	M	0381	1	Opcode_Entry(1, 'BBC', 'ZX'E1', ,branch_1_operand,context_v,context_b),	
250	M	0382	1	Opcode_Entry(1, 'BBCC', 'ZX'E5', ,branch_1_operand,context_v,context_b),	
251	M	0383	1	Opcode_Entry(1, 'BBCCI', 'ZX'E7', ,branch_1_operand,context_v,context_b),	
252	M	0384	1	Opcode_Entry(1, 'BBCS', 'ZX'E3', ,branch_1_operand,context_v,context_b),	
253	M	0385	1	Opcode_Entry(1, 'BBS', 'ZX'E0', ,branch_1_operand,context_v,context_b),	
254	M	0386	1	Opcode_Entry(1, 'BBSC', 'ZX'E4', ,branch_1_operand,context_v,context_b),	
255	M	0387	1	Opcode_Entry(1, 'BBSS', 'ZX'E2', ,branch_1_operand,context_v,context_b),	
256	M	0388	1	Opcode_Entry(1, 'BBSSI', 'ZX'E6', ,branch_1_operand,context_v,context_b),	
257	M	0389	1	Opcode_Entry(0, 'BCC', 'ZX'1E', ,branch_0_operand,context_b),	
258	M	0390	1	Opcode_Entry(0, 'BCS', 'ZX'1F', ,branch_0_operand,context_b),	
259	M	0391	1	Opcode_Entry(1, 'BEQL', 'ZX'13', ,branch_0_operand,context_b),	
260	M	0392	1	Opcode_Entry(0, 'BEQLU', 'ZX'13', ,branch_0_operand,context_b),	
261	M	0393	1	Opcode_Entry(1, 'BGEQ', 'ZX'18', ,branch_0_operand,context_b),	
262	M	0394	1	Opcode_Entry(1, 'BGEQU', 'ZX'1E', ,branch_0_operand,context_b),	
263	M	0395	1	Opcode_Entry(1, 'BGTR', 'ZX'14', ,branch_0_operand,context_b),	
264	M	0396	1	Opcode_Entry(1, 'BGTRU', 'ZX'1A', ,branch_0_operand,context_b),	

```
266 M 0397 1 Opcode_Entry(1, 'BICB2', 'XX'8A', ,simple_2_operand,context_b),
267 M 0398 1 Opcode_Entry(1, 'BICB3', 'XX'8B', ,simple_3_operand,context_b),
268 M 0399 1 Opcode_Entry(1, 'BICL2', 'XX'CA', ,simple_2_operand,context_l),
269 M 0400 1 Opcode_Entry(1, 'BICL3', 'XX'CB', ,simple_3_operand,context_l),
270 M 0401 1 Opcode_Entry(1, 'BICPSW', 'XX'B9', ,simple_1_operand,context_w),
271 M 0402 1 Opcode_Entry(1, 'BICW2', 'XX'AA', ,simple_2_operand,context_w),
272 M 0403 1 Opcode_Entry(1, 'BICW3', 'XX'AB', ,simple_3_operand,context_w),
273 M 0404 1 Opcode_Entry(1, 'BISB2', 'XX'88', ,simple_2_operand,context_b),
274 M 0405 1 Opcode_Entry(1, 'BISB3', 'XX'89', ,simple_3_operand,context_b),
275 M 0406 1 Opcode_Entry(1, 'BISL2', 'XX'C8', ,simple_2_operand,context_l),
276 M 0407 1 Opcode_Entry(1, 'BISL3', 'XX'C9', ,simple_3_operand,context_l),
277 M 0408 1 Opcode_Entry(1, 'BISPSW', 'XX'B8', ,simple_1_operand,context_w),
278 M 0409 1 Opcode_Entry(1, 'BISW2', 'XX'A8', ,simple_2_operand,context_w),
279 M 0410 1 Opcode_Entry(1, 'BISW3', 'XX'A9', ,simple_3_operand,context_w),
280 M 0411 1 Opcode_Entry(1, 'BITB', 'XX'93', ,simple_2_operand,context_b),
281 M 0412 1 Opcode_Entry(1, 'BITL', 'XX'D3', ,simple_2_operand,context_l),
282 M 0413 1 Opcode_Entry(1, 'BITW', 'XX'B3', ,simple_2_operand,context_w),
283 M 0414 1 Opcode_Entry(1, 'BLBC', 'XX'E9', ,branch_1_operand,context_l,context_b),
284 M 0415 1 Opcode_Entry(1, 'BLBS', 'XX'E8', ,branch_1_operand,context_l,context_b),
285 M 0416 1 Opcode_Entry(1, 'BLEQ', 'XX'15', ,branch_0_operand,context_b),
286 M 0417 1 Opcode_Entry(1, 'BLEQU', 'XX'1B', ,branch_0_operand,context_b),
287 M 0418 1 Opcode_Entry(1, 'BLSS', 'XX'19', ,branch_0_operand,context_b),
288 M 0419 1 Opcode_Entry(1, 'BLSSU', 'XX'1F', ,branch_0_operand,context_b),
289 M 0420 1 Opcode_Entry(1, 'BNEQ', 'XX'12', ,branch_0_operand,context_b),
290 M 0421 1 Opcode_Entry(0, 'BNEQU', 'XX'12', ,branch_0_operand,context_b),
291 M 0422 1 Opcode_Entry(1, 'BPT', 'XX'03', ,simple_0_operand),
292 M 0423 1 Opcode_Entry(1, 'BRB', 'XX'11', ,branch_0_operand,context_b),
293 M 0424 1 Opcode_Entry(1, 'BRW', 'XX'31', ,branch_0_operand,context_w),
294 M 0425 1 Opcode_Entry(1, 'BSBB', 'XX'10', ,branch_0_operand,context_b),
295 M 0426 1 Opcode_Entry(1, 'BSBW', 'XX'30', ,branch_0_operand,context_w),
296 M 0427 1 Opcode_Entry(1, 'BUGL', 'XX'FDFF', ,simple_0_operand),
297 M 0428 1 Opcode_Entry(1, 'BUGW', 'XX'FEFF', ,simple_0_operand),
298 M 0429 1 Opcode_Entry(1, 'BVC', 'XX'1C', ,branch_0_operand,context_b),
299 M 0430 1 Opcode_Entry(1, 'BVS', 'XX'1D', ,branch_0_operand,context_b),
300 M 0431 1 Opcode_Entry(1, 'CALLG', 'XX'FA', ,routine_dispatch,context_b,context_b),
301 M 0432 1 Opcode_Entry(1, 'CALLS', 'XX'FB', ,routine_dispatch,context_l,context_b),
302 M 0433 1 Opcode_Entry(1, 'CASEB', 'XX'8F', ,complex_CASE,context_b),
303 M 0434 1 Opcode_Entry(1, 'CASEL', 'XX'CF', ,complex_CASE,context_l),
304 M 0435 1 Opcode_Entry(1, 'CASEW', 'XX'AF', ,complex_CASE,context_w),
305 M 0436 1 Opcode_Entry(1, 'CHME', 'XX'BD', ,simple_1_operand,context_w),
306 M 0437 1 Opcode_Entry(1, 'CHMK', 'XX'BC', ,simple_1_operand,context_w),
307 M 0438 1 Opcode_Entry(1, 'CHMS', 'XX'BE', ,simple_1_operand,context_w),
308 M 0439 1 Opcode_Entry(1, 'CHMU', 'XX'BF', ,simple_1_operand,context_w),
309 M 0440 1 Opcode_Entry(1, 'CLRB', 'XX'94', ,simple_1_operand,context_b),
310 M 0441 1 Opcode_Entry(0, 'CLRD', 'XX'7C', ,simple_1_operand,context_d),
311 M 0442 1 Opcode_Entry(0, 'CLRF', 'XX'D4', ,simple_1_operand,context_f),
312 M 0443 1 Opcode_Entry(0, 'CLRG', 'XX'7C', ,simple_1_operand,context_g),
313 M 0444 1 Opcode_Entry(0, 'CLRH', 'XX'7CFD', ,simple_1_operand,context_h),
314 M 0445 1 Opcode_Entry(1, 'CLRL', 'XX'D4', ,simple_1_operand,context_l),
315 M 0446 1 Opcode_Entry(1, 'CLRO', 'XX'7CFD', ,simple_1_operand,context_o),
316 M 0447 1 Opcode_Entry(1, 'CLRQ', 'XX'7C', ,simple_1_operand,context_q),
317 M 0448 1 Opcode_Entry(1, 'CLRW', 'XX'B4', ,simple_1_operand,context_w),
```



319	M 0449	1	Opcode_Entry(1, 'CMPB	, XX'91'	, simple_2_operand, context_b),
320	M 0450	1	Opcode_Entry(1, 'CMPC3	, XX'29'	, string_3_operand, context_t, context_b),
321	M 0451	1	Opcode_Entry(1, 'CMPC5	, XX'2D'	, string_5_operand, context_t, context_t),
322	M 0452	1	Opcode_Entry(1, 'CMPD	, XX'71'	, simple_2_operand, context_d),
323	M 0453	1	Opcode_Entry(1, 'CMPF	, XX'51'	, simple_2_operand, context_f),
324	M 0454	1	Opcode_Entry(1, 'CMPI	, XX'51FD'	, simple_2_operand, context_g),
325	M 0455	1	Opcode_Entry(1, 'CMPI	, XX'71FD'	, simple_2_operand, context_h),
326	M 0456	1	Opcode_Entry(1, 'CMPL	, XX'D1'	, simple_2_operand, context_l),
327	M 0457	1	Opcode_Entry(1, 'CMPP3	, XX'35'	, string_3_operand, context_p, context_b),
328	M 0458	1	Opcode_Entry(1, 'CMPP4	, XX'37'	, simple_2_operand, context_p),
329	M 0459	1	Opcode_Entry(1, 'CMPV	, XX'EC'	, simple_bit_field, context_m, context_l),
330	M 0460	1	Opcode_Entry(1, 'CMPW	, XX'B1'	, simple_2_operand, context_w),
331	M 0461	1	Opcode_Entry(1, 'CMPZV	, XX'ED'	, simple_bit_field, context_m, context_l),
332	M 0462	1	Opcode_Entry(1, 'CRC	, XX'0B'	, complex_CRC),
333	M 0463	1	Opcode_Entry(1, 'CVTBD	, XX'6C'	, convert_datatype, context_b, context_d),
334	M 0464	1	Opcode_Entry(1, 'CVTBF	, XX'4C'	, convert_datatype, context_b, context_f),
335	M 0465	1	Opcode_Entry(1, 'CVTBG	, XX'4CFD'	, convert_datatype, context_b, context_g),
336	M 0466	1	Opcode_Entry(1, 'CVTBH	, XX'6CFD'	, convert_datatype, context_b, context_h),
337	M 0467	1	Opcode_Entry(1, 'CVTBL	, XX'98'	, convert_datatype, context_b, context_l),
338	M 0468	1	Opcode_Entry(1, 'CVTBW	, XX'99'	, convert_datatype, context_b, context_w),
339	M 0469	1	Opcode_Entry(1, 'CVTDB	, XX'68'	, convert_datatype, context_d, context_b),
340	M 0470	1	Opcode_Entry(1, 'CVTDF	, XX'76'	, convert_datatype, context_d, context_f),
341	M 0471	1	Opcode_Entry(1, 'CVTDH	, XX'32FD'	, convert_datatype, context_d, context_h),
342	M 0472	1	Opcode_Entry(1, 'CVTDL	, XX'6A'	, convert_datatype, context_d, context_l),
343	M 0473	1	Opcode_Entry(1, 'CVTDW	, XX'69'	, convert_datatype, context_d, context_w),
344	M 0474	1	Opcode_Entry(1, 'CVTFB	, XX'48'	, convert_datatype, context_f, context_b),
345	M 0475	1	Opcode_Entry(1, 'CVTFD	, XX'56'	, convert_datatype, context_f, context_d),
346	M 0476	1	Opcode_Entry(1, 'CVTFG	, XX'99FD'	, convert_datatype, context_f, context_g),
347	M 0477	1	Opcode_Entry(1, 'CVTFH	, XX'98FD'	, convert_datatype, context_f, context_h),
348	M 0478	1	Opcode_Entry(1, 'CVTFL	, XX'4A'	, convert_datatype, context_f, context_l),
349	M 0479	1	Opcode_Entry(1, 'CVTFW	, XX'49'	, convert_datatype, context_f, context_w),
350	M 0480	1	Opcode_Entry(1, 'CVTGB	, XX'48FD'	, convert_datatype, context_g, context_b),
351	M 0481	1	Opcode_Entry(1, 'CVTGF	, XX'33FD'	, convert_datatype, context_g, context_f),
352	M 0482	1	Opcode_Entry(1, 'CVTGH	, XX'56FD'	, convert_datatype, context_g, context_h),
353	M 0483	1	Opcode_Entry(1, 'CVTGL	, XX'4AFD'	, convert_datatype, context_g, context_l),
354	M 0484	1	Opcode_Entry(1, 'CVTGW	, XX'49FD'	, convert_datatype, context_g, context_w),
355	M 0485	1	Opcode_Entry(1, 'CVTHB	, XX'68FD'	, convert_datatype, context_h, context_b),
356	M 0486	1	Opcode_Entry(1, 'CVTHD	, XX'F7FD'	, convert_datatype, context_h, context_d),
357	M 0487	1	Opcode_Entry(1, 'CVTHF	, XX'F6FD'	, convert_datatype, context_h, context_f),
358	M 0488	1	Opcode_Entry(1, 'CVTHG	, XX'76FD'	, convert_datatype, context_h, context_g),
359	M 0489	1	Opcode_Entry(1, 'CVTHL	, XX'6AFD'	, convert_datatype, context_h, context_l),
360	M 0490	1	Opcode_Entry(1, 'CVTHW	, XX'69FD'	, convert_datatype, context_h, context_w),
361	M 0491	1	Opcode_Entry(1, 'CVTLB	, XX'F6'	, convert_datatype, context_l, context_b),
362	M 0492	1	Opcode_Entry(1, 'CVTLD	, XX'6E'	, convert_datatype, context_l, context_d),
363	M 0493	1	Opcode_Entry(1, 'CVTLF	, XX'4E'	, convert_datatype, context_l, context_f),
364	M 0494	1	Opcode_Entry(1, 'CVTLG	, XX'4EFD'	, convert_datatype, context_l, context_g),
365	M 0495	1	Opcode_Entry(1, 'CVTLH	, XX'6EFD'	, convert_datatype, context_l, context_h),
366	M 0496	1	Opcode_Entry(1, 'CVTLP	, XX'F9'	, convert_datatype, context_l, context_p),
367	M 0497	1	Opcode_Entry(1, 'CVTLW	, XX'F7'	, convert_datatype, context_l, context_w),



369			Opcode_Entry(1, 'CVTPL', 'XX'36', ,convert_datatype,context_p,context_l),
370			Opcode_Entry(1, 'CVTPS', 'XX'08', ,convert_datatype,context_p,context_t),
371			Opcode_Entry(1, 'CVTPT', 'XX'24', ,string_5_operand,context_p,context_t),
372			Opcode_Entry(1, 'CVTRDL', 'XX'68', ,convert_datatype,context_d,context_l),
373			Opcode_Entry(1, 'CVTRFL', 'XX'48', ,convert_datatype,context_f,context_l),
374			Opcode_Entry(1, 'CVTRGL', 'XX'48FD', ,convert_datatype,context_g,context_l),
375			Opcode_Entry(1, 'CVTRHL', 'XX'68FD', ,convert_datatype,context_h,context_l),
376			Opcode_Entry(1, 'CVTSP', 'XX'09', ,convert_datatype,context_t,context_p),
377			Opcode_Entry(1, 'CVTTP', 'XX'26', ,string_5_operand,context_t,context_p),
378			Opcode_Entry(1, 'CVTWB', 'XX'33', ,convert_datatype,context_w,context_b),
379			Opcode_Entry(1, 'CVTWD', 'XX'6D', ,convert_datatype,context_w,context_d),
380			Opcode_Entry(1, 'CVTWF', 'XX'4D', ,convert_datatype,context_w,context_f),
381			Opcode_Entry(1, 'CVTWG', 'XX'4DFD', ,convert_datatype,context_w,context_g),
382			Opcode_Entry(1, 'CVTWH', 'XX'6DFD', ,convert_datatype,context_w,context_h),
383			Opcode_Entry(1, 'CVTWL', 'XX'32', ,convert_datatype,context_w,context_l),
384			Opcode_Entry(1, 'DECB', 'XX'97', ,simple_1_operand,context_b),
385			Opcode_Entry(1, 'DECL', 'XX'D7', ,simple_1_operand,context_l),
386			Opcode_Entry(1, 'DECW', 'XX'B7', ,simple_1_operand,context_w),
387			Opcode_Entry(1, 'DIVB2', 'XX'86', ,simple_2_operand,context_b),
388			Opcode_Entry(1, 'DIVB3', 'XX'87', ,simple_3_operand,context_b),
389			Opcode_Entry(1, 'DIVD2', 'XX'66', ,simple_2_operand,context_d),
390			Opcode_Entry(1, 'DIVD3', 'XX'67', ,simple_3_operand,context_d),
391			Opcode_Entry(1, 'DIVF2', 'XX'46', ,simple_2_operand,context_f),
392			Opcode_Entry(1, 'DIVF3', 'XX'47', ,simple_3_operand,context_f),
393			Opcode_Entry(1, 'DIVG2', 'XX'46FD', ,simple_2_operand,context_g),
394			Opcode_Entry(1, 'DIVG3', 'XX'47FD', ,simple_3_operand,context_g),
395			Opcode_Entry(1, 'DIVH2', 'XX'66FD', ,simple_2_operand,context_h),
396			Opcode_Entry(1, 'DIVH3', 'XX'67FD', ,simple_3_operand,context_h),
397			Opcode_Entry(1, 'DIVL2', 'XX'C6', ,simple_2_operand,context_l),
398			Opcode_Entry(1, 'DIVL3', 'XX'C7', ,simple_3_operand,context_l),
399			Opcode_Entry(1, 'DIVP', 'XX'27', ,simple_3_operand,context_p),
400			Opcode_Entry(1, 'DIVW2', 'XX'A6', ,simple_2_operand,context_w),
401			Opcode_Entry(1, 'DIVW3', 'XX'A7', ,simple_3_operand,context_w),
402			Opcode_Entry(1, 'EDITPC', 'XX'38', ,string_4_operand,context_p,context_b),
403			Opcode_Entry(1, 'EDIV', 'XX'7B', ,complex_EDIV),
404			Opcode_Entry(1, 'EMODD', 'XX'74', ,complex_EMOD ,context_bu,context_d),
405			Opcode_Entry(1, 'EMODF', 'XX'54', ,complex_EMOD ,context_bu,context_f),
406			Opcode_Entry(1, 'EMODG', 'XX'54FD', ,complex_EMOD ,context_wu,context_g),
407			Opcode_Entry(1, 'EMODH', 'XX'74FD', ,complex_EMOD ,context_wu,context_h),
408			Opcode_Entry(1, 'EMUL', 'XX'7A', ,complex_EMUL),
409			Opcode_Entry(1, 'EXTV', 'XX'EE', ,simple_bit_field,context_m,context_l),
410			Opcode_Entry(1, 'EXTZV', 'XX'EF', ,simple_bit_field,context_m,context_l),
411			Opcode_Entry(1, 'FFC', 'XX'EB', ,simple_bit_field,context_m,context_l),
412			Opcode_Entry(1, 'FFS', 'XX'EA', ,simple_bit_field,context_m,context_l),
413			Opcode_Entry(1, 'HALT', 'XX'00', ,simple_0_operand),
414			Opcode_Entry(1, 'INCB', 'XX'96', ,simple_1_operand,context_b),
415			Opcode_Entry(1, 'INCL', 'XX'D6', ,simple_1_operand,context_l),
416			Opcode_Entry(1, 'INCW', 'XX'B6', ,simple_1_operand,context_w),
417			Opcode_Entry(1, 'INDEX', 'XX'0A', ,complex_INDEX),

419	M	0547	1	Opcode_Entry(1, 'INSQHI', 'XX'5C',	, simple_2_operand, context_b),
420	M	0548	1	Opcode_Entry(1, 'INSQTI', 'XX'5D',	, simple_2_operand, context_b),
421	M	0549	1	Opcode_Entry(1, 'INSQUE', 'XX'0E',	, simple_2_operand, context_b),
422	M	0550	1	Opcode_Entry(1, 'INSV', 'XX'F0',	, simple_bit_field, context_l, context_m),
423	M	0551	1	Opcode_Entry(1, 'JMP', 'XX'17',	, simple_1_operand, context_b),
424	M	0552	1	Opcode_Entry(1, 'JSB', 'XX'16',	, simple_1_operand, context_b),
425	M	0553	1	Opcode_Entry(1, 'LDPCTX', 'XX'06',	, simple_0_operand),
426	M	0554	1	Opcode_Entry(1, 'LOCC', 'XX'3A',	, locate_character, context_b, context_t),
427	M	0555	1	Opcode_Entry(1, 'MATCHC', 'XX'39',	, simple_2_operand, context_t),
428	M	0556	1	Opcode_Entry(1, 'MCOMB', 'XX'92',	, simple_2_operand, context_b),
429	M	0557	1	Opcode_Entry(1, 'MCOML', 'XX'D2',	, simple_2_operand, context_l),
430	M	0558	1	Opcode_Entry(1, 'MCOMW', 'XX'B2',	, simple_2_operand, context_w),
431	M	0559	1	Opcode_Entry(1, 'MFPB', 'XX'DB',	, simple_2_operand, context_l),
432	M	0560	1	Opcode_Entry(1, 'MNEGB', 'XX'8E',	, simple_2_operand, context_b),
433	M	0561	1	Opcode_Entry(1, 'MNEGD', 'XX'72',	, simple_2_operand, context_d),
434	M	0562	1	Opcode_Entry(1, 'MNEGF', 'XX'52',	, simple_2_operand, context_f),
435	M	0563	1	Opcode_Entry(1, 'MNEGG', 'XX'52FD',	, simple_2_operand, context_g),
436	M	0564	1	Opcode_Entry(1, 'MNEGH', 'XX'72FD',	, simple_2_operand, context_h),
437	M	0565	1	Opcode_Entry(1, 'MNEGL', 'XX'CE',	, simple_2_operand, context_l),
438	M	0566	1	Opcode_Entry(1, 'MNEGW', 'XX'AE',	, simple_2_operand, context_w),
439	M	0567	1	Opcode_Entry(1, 'MOVAB', 'XX'9E',	, evaluate_address, context_b, context_l),
440	M	0568	1	Opcode_Entry(0, 'MOVAD', 'XX'7E',	, evaluate_address, context_d, context_l),
441	M	0569	1	Opcode_Entry(0, 'MOVAF', 'XX'DE',	, evaluate_address, context_f, context_l),
442	M	0570	1	Opcode_Entry(0, 'MOVAG', 'XX'7E',	, evaluate_address, context_g, context_l),
443	M	0571	1	Opcode_Entry(0, 'MOVAH', 'XX'7EFD',	, evaluate_address, context_h, context_l),
444	M	0572	1	Opcode_Entry(1, 'MOVAL', 'XX'DE',	, evaluate_address, context_l, context_l),
445	M	0573	1	Opcode_Entry(1, 'MOVAO', 'XX'7EFD',	, evaluate_address, context_o, context_l),
446	M	0574	1	Opcode_Entry(1, 'MOVAQ', 'XX'7E',	, evaluate_address, context_q, context_l),
447	M	0575	1	Opcode_Entry(1, 'MOVAV', 'XX'3E',	, evaluate_address, context_v, context_l),
448	M	0576	1	Opcode_Entry(1, 'MOVB', 'XX'90',	, simple_2_operand, context_b),
449	M	0577	1	Opcode_Entry(1, 'MOVCB', 'XX'28',	, string_3_operand, context_t, context_b),
450	M	0578	1	Opcode_Entry(1, 'MOVCS', 'XX'2C',	, string_3_operand, context_t, context_t),
451	M	0579	1	Opcode_Entry(1, 'MOVD', 'XX'70',	, simple_2_operand, context_d),
452	M	0580	1	Opcode_Entry(1, 'MOVF', 'XX'50',	, simple_2_operand, context_f),
453	M	0581	1	Opcode_Entry(1, 'MOVG', 'XX'50FD',	, simple_2_operand, context_g),
454	M	0582	1	Opcode_Entry(1, 'MOVH', 'XX'70FD',	, simple_2_operand, context_h),
455	M	0583	1	Opcode_Entry(1, 'MOVL', 'XX'D0',	, simple_2_operand, context_l),
456	M	0584	1	Opcode_Entry(1, 'MOVQ', 'XX'7DFD',	, simple_2_operand, context_o),
457	M	0585	1	Opcode_Entry(1, 'MOVP', 'XX'34',	, string_3_operand, context_p, context_b),
458	M	0586	1	Opcode_Entry(1, 'MOVPSL', 'XX'DC',	, simple_1_operand, context_l),
459	M	0587	1	Opcode_Entry(1, 'MOVQ', 'XX'7D',	, simple_2_operand, context_q),
460	M	0588	1	Opcode_Entry(1, 'MOVTC', 'XX'2E',	, string_6_operand, context_t, context_t),
461	M	0589	1	Opcode_Entry(1, 'MOVTC', 'XX'2F',	, string_6_operand, context_t, context_t),
462	M	0590	1	Opcode_Entry(1, 'MOVW', 'XX'80',	, simple_2_operand, context_w),
463	M	0591	1	Opcode_Entry(1, 'MOVZBL', 'XX'9A',	, convert_datatype, context_b, context_l),
464	M	0592	1	Opcode_Entry(1, 'MOVZBW', 'XX'9B',	, convert_datatype, context_b, context_w),
465	M	0593	1	Opcode_Entry(1, 'MOVZWL', 'XX'3C',	, convert_datatype, context_w, context_l),
466	M	0594	1	Opcode_Entry(1, 'MTPR', 'XX'DA',	, simple_2_operand, context_l),

```
468      M 0595 1 Opcode_Entry(1, 'MULB2', 'XX'84', .simple_2_operand, context_b),
469      M 0596 1 Opcode_Entry(1, 'MULB3', 'XX'85', .simple_3_operand, context_b),
470      M 0597 1 Opcode_Entry(1, 'MULD2', 'XX'64', .simple_2_operand, context_d),
471      M 0598 1 Opcode_Entry(1, 'MULD3', 'XX'65', .simple_3_operand, context_d),
472      M 0599 1 Opcode_Entry(1, 'MULF2', 'XX'44', .simple_2_operand, context_f),
473      M 0600 1 Opcode_Entry(1, 'MULF3', 'XX'45', .simple_3_operand, context_f),
474      M 0601 1 Opcode_Entry(1, 'MULG2', 'XX'44FD', .simple_2_operand, context_g),
475      M 0602 1 Opcode_Entry(1, 'MULG3', 'XX'45FD', .simple_3_operand, context_g),
476      M 0603 1 Opcode_Entry(1, 'MULH2', 'XX'64FD', .simple_2_operand, context_h),
477      M 0604 1 Opcode_Entry(1, 'MULH3', 'XX'65FD', .simple_3_operand, context_h),
478      M 0605 1 Opcode_Entry(1, 'MULL2', 'XX'C4', .simple_2_operand, context_l),
479      M 0606 1 Opcode_Entry(1, 'MULL3', 'XX'C5', .simple_3_operand, context_l),
480      M 0607 1 Opcode_Entry(1, 'MULP', 'XX'25', .simple_3_operand, context_p),
481      M 0608 1 Opcode_Entry(1, 'MULW2', 'XX'A4', .simple_2_operand, context_w),
482      M 0609 1 Opcode_Entry(1, 'MULW3', 'XX'A5', .simple_3_operand, context_w),
483      M 0610 1 Opcode_Entry(1, 'NOP', 'XX'01', .simple_0_operand),
484      M 0611 1 Opcode_Entry(1, 'POLYD', 'XX'75', .polynomial_value, context_d, context_t),
485      M 0612 1 Opcode_Entry(1, 'POLYF', 'XX'55', .polynomial_value, context_f, context_t),
486      M 0613 1 Opcode_Entry(1, 'POLYG', 'XX'55FD', .polynomial_value, context_g, context_t),
487      M 0614 1 Opcode_Entry(1, 'POLYH', 'XX'75FD', .polynomial_value, context_h, context_t),
488      M 0615 1 Opcode_Entry(1, 'POPR', 'XX'BA', .simple_1_operand, context_w),
489      M 0616 1 Opcode_Entry(1, 'PROBER', 'XX'0C', .probe_for_access, context_b, context_t),
490      M 0617 1 Opcode_Entry(1, 'PROBEW', 'XX'0D', .probe_for_access, context_b, context_t),
491      M 0618 1 Opcode_Entry(1, 'PUSHAB', 'XX'9F', .simple_1_operand, context_b),
492      M 0619 1 Opcode_Entry(0, 'PUSHAD', 'XX'7F', .simple_1_operand, context_d),
493      M 0620 1 Opcode_Entry(0, 'PUSHAF', 'XX'DF', .simple_1_operand, context_f),
494      M 0621 1 Opcode_Entry(0, 'PUSHAG', 'XX'7F', .simple_1_operand, context_g),
495      M 0622 1 Opcode_Entry(0, 'PUSHAH', 'XX'7FFD', .simple_1_operand, context_h),
496      M 0623 1 Opcode_Entry(1, 'PUSHAL', 'XX'DF', .simple_1_operand, context_l),
497      M 0624 1 Opcode_Entry(1, 'PUSHAO', 'XX'7FFD', .simple_1_operand, context_o),
498      M 0625 1 Opcode_Entry(1, 'PUSHAQ', 'XX'7F', .simple_1_operand, context_q),
499      M 0626 1 Opcode_Entry(1, 'PUSHAW', 'XX'3F', .simple_1_operand, context_w),
500      M 0627 1 Opcode_Entry(1, 'PUSHL', 'XX'DD', .simple_1_operand, context_l),
501      M 0628 1 Opcode_Entry(1, 'PUSHR', 'XX'BB', .simple_1_operand, context_w),
502      M 0629 1 Opcode_Entry(1, 'REI', 'XX'02', .simple_0_operand),
503      M 0630 1 Opcode_Entry(1, 'REMQHI', 'XX'5E', .simple_2_operand, context_b),
504      M 0631 1 Opcode_Entry(1, 'REMQTI', 'XX'5F', .simple_2_operand, context_b),
505      M 0632 1 Opcode_Entry(1, 'REMQUE', 'XX'0F', .simple_2_operand, context_b),
506      M 0633 1 Opcode_Entry(1, 'RET', 'XX'04', .simple_0_operand),
507      M 0634 1 Opcode_Entry(1, 'ROTL', 'XX'9C', .complex_SHIFT, context_l),
508      M 0635 1 Opcode_Entry(1, 'RSB', 'XX'05', .simple_0_operand),
509      M 0636 1 Opcode_Entry(1, 'SBWC', 'XX'D9', .simple_2_operand, context_l),
510      M 0637 1 Opcode_Entry(1, 'SCANC', 'XX'2A', .string_4_operand, context_t, context_b),
511      M 0638 1 Opcode_Entry(1, 'SKPC', 'XX'3B', .locate_character, context_b, context_t),
512      M 0639 1 Opcode_Entry(1, 'SOBGEO', 'XX'F4', .branch_1_operand, context_l, context_b),
513      M 0640 1 Opcode_Entry(1, 'SOBGTR', 'XX'F5', .branch_1_operand, context_l, context_b),
514      M 0641 1 Opcode_Entry(1, 'SPANC', 'XX'2B', .string_4_operand, context_t, context_b),
```



```

... 516      0642      1      Opcode_Entry(1,'SUBB2',.XX'82',.simple_2_operand,context_b),
517      0643      1      Opcode_Entry(1,'SUBB3',.XX'83',.simple_3_operand,context_b),
518      0644      1      Opcode_Entry(1,'SUBD2',.XX'62',.simple_2_operand,context_d),
519      0645      1      Opcode_Entry(1,'SUBD3',.XX'63',.simple_3_operand,context_d),
520      0646      1      Opcode_Entry(1,'SUBF2',.XX'42',.simple_2_operand,context_f),
521      0647      1      Opcode_Entry(1,'SUBF3',.XX'43',.simple_3_operand,context_f),
522      0648      1      Opcode_Entry(1,'SUBG2',.XX'42FD',.simple_2_operand,context_g),
523      0649      1      Opcode_Entry(1,'SUBG3',.XX'43FD',.simple_3_operand,context_g),
524      0650      1      Opcode_Entry(1,'SUBH2',.XX'62FD',.simple_2_operand,context_h),
525      0651      1      Opcode_Entry(1,'SUBH3',.XX'63FD',.simple_3_operand,context_h),
526      0652      1      Opcode_Entry(1,'SUBL2',.XX'C2',.simple_2_operand,context_l),
527      0653      1      Opcode_Entry(1,'SUBL3',.XX'C3',.simple_3_operand,context_l),
528      0654      1      Opcode_Entry(1,'SUBP4',.XX'22',.simple_2_operand,context_p),
529      0655      1      Opcode_Entry(1,'SUBP6',.XX'23',.simple_3_operand,context_p),
530      0656      1      Opcode_Entry(1,'SUBW2',.XX'A2',.simple_2_operand,context_w),
531      0657      1      Opcode_Entry(1,'SUBW3',.XX'A3',.simple_3_operand,context_w),
532      0658      1      Opcode_Entry(1,'SVPCTX',.XX'07',.simple_0_operand),
533      0659      1      Opcode_Entry(1,'TSTB',.XX'95',.simple_1_operand,context_b),
534      0660      1      Opcode_Entry(1,'TSTD',.XX'73',.simple_1_operand,context_d),
535      0661      1      Opcode_Entry(1,'TSTF',.XX'53',.simple_1_operand,context_f),
536      0662      1      Opcode_Entry(1,'TSTG',.XX'53FD',.simple_1_operand,context_g),
537      0663      1      Opcode_Entry(1,'TSTH',.XX'73FD',.simple_1_operand,context_h),
538      0664      1      Opcode_Entry(1,'TSTL',.XX'D5',.simple_1_operand,context_l),
539      0665      1      Opcode_Entry(1,'TSTW',.XX'B5',.simple_1_operand,context_w),
540      0666      1      Opcode_Entry(1,'XFC',.XX'FC',.simple_0_operand),
541      0667      1      Opcode_Entry(1,'XORB2',.XX'8C',.simple_2_operand,context_b),
542      0668      1      Opcode_Entry(1,'XORB3',.XX'8D',.simple_3_operand,context_b),
543      0669      1      Opcode_Entry(1,'XORL2',.XX'CC',.simple_2_operand,context_l),
544      0670      1      Opcode_Entry(1,'XORL3',.XX'CD',.simple_3_operand,context_l),
545      0671      1      Opcode_Entry(1,'XORW2',.XX'AC',.simple_2_operand,context_w),
546      0672      1      Opcode_Entry(1,'XORW3',.XX'AD',.simple_3_operand,context_w),
547      0673      1
548      0674      1  %;

```

! \*\*\*\* End of definition of MACRO Opcode\_List

```
... 550 0675 1 |++
551 0676 1 |
552 0677 1 |     Used in the macro Opcode_List which is used in the following
553 0678 1 |     GLOBAL BIND to define DBG$Opcode_Name_table
554 0679 1 |
555 0680 1 | --
556 0681 1 | MACRO Opcode_Entry(Flag, Name, Code, Kind, Arg1, Arg2) =
557 0682 1 |     BYTE(ASCII Name),
558 0683 1 |     WORD(XIF ((Code) GTRU XX'FF') XTHEN (Code) XELSE (Code)^8 XFI),
559 0684 1 |     BYTE(Kind),
560 0685 1 |     BYTE(XIF (XLENGTH LSS 5) XTHEN 0 XELSE
561 0686 1 |         XIF (XLENGTH EQL 5) XTHEN (Arg1) XELSE (((Arg1)^4)+(Arg2))
562 0687 1 |         XFI XFI) X;
563 0688 1 |
564 0689 1 | GLOBAL BIND
565 0690 1 |     DBG$Opcode_Name_Table =
566 0691 1 |         UPLIT WORD(BYTE(ASCII '??????'),WORD(0,0),Opcode_List) : BLOCKVECTOR [,10,BYTE];
567 0692 1 |
568 0693 1 | ++
569 0694 1 |
570 0695 1 |     Undeclare Opcode_Entry so that another definition of the macro
571 0696 1 |     Opcode_Entry can be defined to build the back translation
572 0697 1 |     table, DBG$Opcode_Kind_Table.
573 0698 1 | --
574 0699 1 | UNDECLARE %QUOTE Opcode_Entry;
575 0700 1 |
576 0701 1 | MACRO Opcode_Entry(Flag, Name, Code, Kind, Arg1, Arg2) =
577 0702 1 |     %ASSIGN(Opcode_Index, Opcode_Index + 1)
578 0703 1 |     %IF Flag
579 0704 1 |     %THEN
580 0705 1 |         [IF (Code LSS XX'100') THEN (Code) ELSE (((Code)^-8)+XX'100')] = Opcode_Index
581 0706 1 |     %ELSE
582 0707 1 |     %ASSIGN(Ignore_Index, Ignore_Index + 1)
583 0708 1 |         [255 + Ignore_Index] = 0
584 0709 1 |     %FI
585 0710 1 |     X;
586 0711 1 |
587 0712 1 | COMPILETIME
588 0713 1 |     Opcode_Index = 0,
589 0714 1 |     Ignore_Index = 0;
590 0715 1 |
591 0716 1 | GLOBAL
592 0717 1 |     DBG$Opcode_Kind_Table : PSECT(DBG$PLIT) VECTOR[512,WORD] PRESET(Opcode_List);
593 0718 1 |
594 0719 1 | UNDECLARE %QUOTE Opcode_Entry;
595 0720 1 |
596 0721 1 | LITERAL Opcode_Table_Size = Opcode_Index;
```

```
598 0722 1 OWN
599 0723 1 Op Buffer : BLOCK [16, BYTE],
600 0724 1 Data Size : PSECT(DBG$PLIT) VECTOR[12, BYTE] PRESET(
601 0725 1 [context_b] = 1, [context_w] = 2;
602 0726 1 [context_l] = 4, [context_q] = 8;
603 0727 1 [context_o] = 16,
604 0728 1 [context_f] = 4, [context_d] = 8;
605 0729 1 [context_g] = 8, [context_h] = 16;
606 0730 1 [context_bu] = 1, [context_wu] = 2;
607 0731 1 Data Type : PSECT(DBG$PLIT) VECTOR[12, BYTE] PRESET(
608 0732 1 [context_b] = dsc$sk_dtype_b, [context_w] = dsc$sk_dtype_w,
609 0733 1 [context_l] = dsc$sk_dtype_l, [context_q] = dsc$sk_dtype_q,
610 0734 1 [context_o] = dsc$sk_dtype_o,
611 0735 1 [context_f] = dsc$sk_dtype_f, [context_d] = dsc$sk_dtype_d,
612 0736 1 [context_g] = dsc$sk_dtype_g, [context_h] = dsc$sk_dtype_h,
613 0737 1 [context_bu] = dsc$sk_dtype_bu, [context_wu] = dsc$sk_dtype_wu);
614 0738 1
615 0739 1 BIND
616 0740 1 Operand Value = Op Buffer : LONG,
617 0741 1 Register Name = UPLIT BYTE(
618 0742 1 2, 'R', '0', 0, 2, 'R', '1', 0, 2, 'R', '2', 0, 2, 'R', '3', 0,
619 0743 1 2, 'R', '4', 0, 2, 'R', '5', 0, 2, 'R', '6', 0, 2, 'R', '7', 0,
620 0744 1 2, 'R', '8', 0, 2, 'R', '9', 0, 3, 'R', '1', 1,
621 0745 1 2, 'A', 'P', 0, 2, 'F', 'P', 0, 2, 'S', 'P', 0, 2, 'P', 'C', 0,
622 0746 1 !
623 0747 1 2, '?', '?', 0, 2, '?', '?', 0, 2, 'I', 'V', 0, 2, 'D', 'V', 0 )
624 0748 1 : VECTOR[.LONG],
625 0749 1 Format_AD = UPLIT BYTE(%ASCIC '!AD'),
626 0750 1 Format_AC = UPLIT BYTE(%ASCIC '!AC'),
627 0751 1 comma = UPLIT BYTE(',');
628 0752 1
629 0753 1 MACRO
630 0754 1 offset = 0.0, 0.0 %;
631 0755 1 u_byte = 0.0, 8.0 %;
632 0756 1 u_word = 0.0, 16.0 %;
633 0757 1 u_long = 0.0, 32.0 %;
634 0758 1 s_byte = 0.0, 8.1 %;
635 0759 1 s_word = 0.0, 16.1 %;
636 0760 1 s_long = 0.0, 32.1 %;
637 0761 1
638 0762 1 FIELD Opcode_Entry_Fields =
639 0763 1 SET
640 0764 1 op_name = [0.0, 0.0],
641 0765 1 op_code = [6.0, 16.0],
642 0766 1 op_code_one = [6.0, 8.0],
643 0767 1 op_code_two = [7.0, 8.0],
644 0768 1 op_kind = [8.0, 8.0],
645 0769 1 op_type = [9.0, 8.0],
646 0770 1 op_type_one = [9.0, 4.0],
647 0771 1 op_type_two = [9.4, 4.0],
648 0772 1 TES;
649 0773 1
650 0774 1 FIELD Encode_Fields =
651 0775 1 SET
652 0776 1 Enc_Input_Desc = [ 0, 0, 0.0],
653 0777 1 Enc_Input_Length = [ 0, 0, 16.0],
654 0778 1 Enc_Input_Dtype = [ 2, 0, 8.0]
```

! Whole opcode



DBGENCDEC  
V04-000

13  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 15  
(10)

..	655	0779		Enc_Input_Class	=	[3. 0. 8. 0].
..	656	0780		Enc_Input_Buffer	=	[4. 0. 32. 0].
..	657	0781		Enc_Output_Length	=	[8. 0. 32. 0].
..	658	0782		Enc_Output_Buffer	=	[12. 0. 32. 0].
..	659	0783		Enc_Final_Address	=	[16. 0. 32. 0].
..	660	0784		TES:		

```

662 0785 1 GLOBAL ROUTINE DBGSins_Decode(Start_Address,Print_Flag,Entry_Flag) =
663 0786 BEGIN
664 0787 BUILTIN ACTUALCOUNT;
665 0788 LOCAL Pointer;
666 0789 LOCAL
667 0790 Signal_Flag : VOLATILE,
668 0791 Error_Value : VOLATILE;
669 0792
670 0793 ROUTINE Decode_Handler(Sig_Args,Mech_Args,Enable_Args) =
671 0794 BEGIN
672 0795 MAP Sig_Args : REF BLOCK[,BYTE],
673 0796 Mech_Args : REF BLOCK[,BYTE],
674 0797 Enable_Args : REF VECTOR[,LONG];
675 0798 EXTERNAL ROUTINE SYSSUNWIND : Addressing_Mode(General);
676 0799
677 0800 IF .Sig_Args[chf$!_sig_name] EQL ss$ unwind THEN RETURN ss$ continue;
678 0801 IF .(.Enable_Args[T]) THEN RETURN ss$ resignal;
679 0802
680 0803 Mech_Args[chf$!_mch_savr0] = .(.Enable_Args[2]);
681 0804
682 0805 SYSSUNWIND(0,0);
683 0806 RETURN ss$ continue;
684 0807 END;

```

						.TITLE	DBGENCDEC
						.IDENT	\V04-000\
						.PSECT	DBG\$PLIT,NOWRT, SHR, PIC,0
3F	3F	3F	3F	3F	3F	00000	P.AAA: .ASCII \??????\
			0000	0000	0000	00006	.WORD 0,0
20	20	42	42	43	41	0000A	.ASCII \ACBB \
				9D00		00010	.WORD -25344
				07		00012	.BYTE 7
				01		00013	.BYTE 1
20	20	44	42	43	41	00014	.ASCII \ACBD \
				6F00		0001A	.WORD 28416
				07		0001C	.BYTE 7
				61		0001D	.BYTE 97
20	20	46	42	43	41	0001E	.ASCII \ACBF \
				4F00		00024	.WORD 20224
				07		00026	.BYTE 7
				51		00027	.BYTE 81
20	20	47	42	43	41	00028	.ASCII \ACBG \
				4FFD		0002E	.WORD 20477
				07		00030	.BYTE 7
				71		00031	.BYTE 113
20	20	48	42	43	41	00032	.ASCII \ACBH \
				6FFD		00038	.WORD 28669
				07		0003A	.BYTE 7
				81		0003B	.BYTE -127
20	20	4C	42	43	41	0003C	.ASCII \ACBL \
				F100		00042	.WORD -3840
				07		00044	.BYTE 7
				21		00045	.BYTE 33
20	20	57	42	43	41	00046	.ASCII \ACBW \

					3D00	0004C	.WORD	15616
					07	0004E	.BYTE	7
					11	0004F	.BYTE	17
20	49	57	41	44	41	00050	.ASCII	\ADAW1 \
					5800	00056	.WORD	22528
					02	00058	.BYTE	2
					01	00059	.BYTE	1
20	32	42	44	44	41	0005A	.ASCII	\ADDB2 \
					8000	00060	.WORD	-32768
					02	00062	.BYTE	2
					00	00063	.BYTE	0
20	33	42	44	44	41	00064	.ASCII	\ADDB3 \
					8100	0006A	.WORD	-32512
					03	0006C	.BYTE	3
					00	0006D	.BYTE	0
20	32	44	44	44	41	0006E	.ASCII	\ADDD2 \
					6000	00074	.WORD	24576
					02	00076	.BYTE	2
					06	00077	.BYTE	6
20	33	44	44	44	41	00078	.ASCII	\ADDD3 \
					6100	0007E	.WORD	24832
					03	00080	.BYTE	3
					06	00081	.BYTE	6
20	32	46	44	44	41	00082	.ASCII	\ADDF2 \
					4000	00088	.WORD	16384
					02	0008A	.BYTE	2
					05	0008B	.BYTE	5
20	33	46	44	44	41	0008C	.ASCII	\ADDF3 \
					4100	00092	.WORD	16640
					03	00094	.BYTE	3
					05	00095	.BYTE	5
20	32	47	44	44	41	00096	.ASCII	\ADDG2 \
					40FD	0009C	.WORD	16637
					02	0009E	.BYTE	2
					07	0009F	.BYTE	7
20	33	47	44	44	41	000A0	.ASCII	\ADDG3 \
					41FD	000A6	.WORD	16893
					03	000A8	.BYTE	3
					07	000A9	.BYTE	7
20	32	48	44	44	41	000AA	.ASCII	\ADDH2 \
					60FD	000B0	.WORD	24829
					02	000B2	.BYTE	2
					08	000B3	.BYTE	8
20	33	48	44	44	41	000B4	.ASCII	\ADDH3 \
					61FD	000BA	.WORD	25085
					03	000BC	.BYTE	3
					08	000BD	.BYTE	8
20	32	4C	44	44	41	000BE	.ASCII	\ADDL2 \
					C000	000C4	.WORD	-16384
					02	000C6	.BYTE	2
					02	000C7	.BYTE	2
20	33	4C	44	44	41	000C8	.ASCII	\ADDL3 \
					C100	000CE	.WORD	-16128
					03	000D0	.BYTE	3
					02	000D1	.BYTE	2
20	34	50	44	44	41	000D2	.ASCII	\ADDP4 \
					2000	000D8	.WORD	8192



					02	000DA	.BYTE	2	
					0C	000DB	.BYTE	12	
20	36	50	44	44	41	000DC	.ASCII	\ADDP6 \	
					2100	000E2	.WORD	8448	
					03	000E4	.BYTE	3	
					0C	000E5	.BYTE	12	
20	32	57	44	44	41	000E6	.ASCII	\ADDW2 \	
					A000	000EC	.WORD	-24576	
					02	000EE	.BYTE	2	
					01	000EF	.BYTE	1	
20	33	57	44	44	41	000F0	.ASCII	\ADDW3 \	
					A100	000F6	.WORD	-24320	
					03	000F8	.BYTE	3	
					01	000F9	.BYTE	1	
20	20	43	57	44	41	000FA	.ASCII	\ADWC \	
					D800	00100	.WORD	-10240	
					02	00102	.BYTE	2	
					02	00103	.BYTE	2	
51	45	4C	42	4F	41	00104	.ASCII	\AOBLEQ\	
					F300	0010A	.WORD	-3328	
					06	0010C	.BYTE	6	
					20	0010D	.BYTE	32	
53	53	4C	42	4F	41	0010E	.ASCII	\AOBLSS\	
					F200	00114	.WORD	-3584	
					06	00116	.BYTE	6	
					20	00117	.BYTE	32	
20	20	4C	48	53	41	00118	.ASCII	\ASHL \	
					7800	0011E	.WORD	30720	
					10	00120	.BYTE	16	
					02	00121	.BYTE	2	
20	20	50	48	53	41	00122	.ASCII	\ASHP \	
					F800	00128	.WORD	-2048	
					17	0012A	.BYTE	23	
					0C	0012B	.BYTE	12	
20	20	51	48	53	41	0012C	.ASCII	\ASHQ \	
					7900	00132	.WORD	30976	
					10	00134	.BYTE	16	
					03	00135	.BYTE	3	
20	20	20	43	42	42	00136	.ASCII	\BBC \	
					E100	0013C	.WORD	-7936	
					05	0013E	.BYTE	5	
					E0	0013F	.BYTE	-32	
20	20	43	43	42	42	00140	.ASCII	\BBCC \	
					E500	00146	.WORD	-6912	
					05	00148	.BYTE	5	
					E0	00149	.BYTE	-32	
20	49	43	43	42	42	0014A	.ASCII	\BBCCI \	
					E700	00150	.WORD	-6400	
					05	00152	.BYTE	5	
					E0	00153	.BYTE	-32	
20	20	53	43	42	42	00154	.ASCII	\BBCS \	
					E300	0015A	.WORD	-7424	
					05	0015C	.BYTE	5	
					E0	0015D	.BYTE	-32	
20	20	20	53	42	42	0015E	.ASCII	\BBS \	
					E000	00164	.WORD	-8192	
					05	00166	.BYTE	5	

					E0	00167	.BYTE	-32	
20	20	43	53	42	42	00168	.ASCII	\BBSC \	
					E400	0016E	.WORD	-7168	
					05	00170	.BYTE	5	
					E0	00171	.BYTE	-32	
20	20	53	53	42	42	00172	.ASCII	\BBSS \	
					E200	00178	.WORD	-7680	
					05	0017A	.BYTE	5	
					E0	0017B	.BYTE	-32	
20	49	53	53	42	42	0017C	.ASCII	\BBSSI \	
					E600	00182	.WORD	-6656	
					05	00184	.BYTE	5	
					E0	00185	.BYTE	-32	
20	20	20	43	43	42	00186	.ASCII	\BCC \	
					1E00	0018C	.WORD	7680	
					04	0018E	.BYTE	4	
					00	0018F	.BYTE	0	
20	20	20	53	43	42	00190	.ASCII	\BCS \	
					1F00	00196	.WORD	7936	
					04	00198	.BYTE	4	
					00	00199	.BYTE	0	
20	20	4C	51	45	42	0019A	.ASCII	\BEQL \	
					1300	001A0	.WORD	4864	
					04	001A2	.BYTE	4	
					00	001A3	.BYTE	0	
20	55	4C	51	45	42	001A4	.ASCII	\BEQLU \	
					1300	001AA	.WORD	4864	
					04	001AC	.BYTE	4	
					00	001AD	.BYTE	0	
20	20	51	45	47	42	001AE	.ASCII	\BGEQ \	
					1800	001B4	.WORD	6144	
					04	001B6	.BYTE	4	
					00	001B7	.BYTE	0	
20	55	51	45	47	42	001B8	.ASCII	\BGEQU \	
					1E00	001BE	.WORD	7680	
					04	001C0	.BYTE	4	
					00	001C1	.BYTE	0	
20	20	52	54	47	42	001C2	.ASCII	\BGTR \	
					1400	001C8	.WORD	5120	
					04	001CA	.BYTE	4	
					00	001CB	.BYTE	0	
20	55	52	54	47	42	001CC	.ASCII	\BGTRU \	
					1A00	001D2	.WORD	6656	
					04	001D4	.BYTE	4	
					00	001D5	.BYTE	0	
20	32	42	43	49	42	001D6	.ASCII	\BICB2 \	
					8A00	001DC	.WORD	-30208	
					02	001DE	.BYTE	2	
					00	001DF	.BYTE	0	
20	33	42	43	49	42	001E0	.ASCII	\BICB3 \	
					8B00	001E6	.WORD	-29952	
					03	001E8	.BYTE	3	
					00	001E9	.BYTE	0	
20	32	4C	43	49	42	001EA	.ASCII	\BICL2 \	
					CA00	001F0	.WORD	-13824	
					02	001F2	.BYTE	2	
					02	001F3	.BYTE	2	

20	33	4C	43	49	42	001F4	.ASCII	\BICL3 \
					CB00	001FA	.WORD	-13568
					03	001FC	.BYTE	3
					02	001FD	.BYTE	2
57	53	50	43	49	42	001FE	.ASCII	\BICPSW\
					B900	00204	.WORD	-18176
					01	00206	.BYTE	1
					01	00207	.BYTE	1
20	32	57	43	49	42	00208	.ASCII	\BICW2 \
					AA00	0020E	.WORD	-22016
					02	00210	.BYTE	2
					01	00211	.BYTE	1
20	33	57	43	49	42	00212	.ASCII	\BICW3 \
					AB00	00218	.WORD	-21760
					03	0021A	.BYTE	3
					01	0021B	.BYTE	1
20	32	42	53	49	42	0021C	.ASCII	\BISB2 \
					B800	00222	.WORD	-30720
					02	00224	.BYTE	2
					00	00225	.BYTE	0
20	33	42	53	49	42	00226	.ASCII	\BISB3 \
					B900	0022C	.WORD	-30464
					03	0022E	.BYTE	3
					00	0022F	.BYTE	0
20	32	4C	53	49	42	00230	.ASCII	\BISL2 \
					C800	00236	.WORD	-14336
					02	00238	.BYTE	2
					02	00239	.BYTE	2
20	33	4C	53	49	42	0023A	.ASCII	\BISL3 \
					C900	00240	.WORD	-14080
					03	00242	.BYTE	3
					02	00243	.BYTE	2
57	53	50	53	49	42	00244	.ASCII	\BISPSW\
					B800	0024A	.WORD	-18432
					01	0024C	.BYTE	1
					01	0024D	.BYTE	1
20	32	57	53	49	42	0024E	.ASCII	\BISW2 \
					A800	00254	.WORD	-22528
					02	00256	.BYTE	2
					01	00257	.BYTE	1
20	33	57	53	49	42	00258	.ASCII	\BISW3 \
					A900	0025E	.WORD	-22272
					03	00260	.BYTE	3
					01	00261	.BYTE	1
20	20	42	54	49	42	00262	.ASCII	\BITB \
					9300	00268	.WORD	-27904
					02	0026A	.BYTE	2
					00	0026B	.BYTE	0
20	20	4C	54	49	42	0026C	.ASCII	\BITL \
					D300	00272	.WORD	-11520
					02	00274	.BYTE	2
					03	00275	.BYTE	2
20	20	57	54	49	42	00276	.ASCII	\BITW \
					B300	0027C	.WORD	-19712
					02	0027E	.BYTE	2
					01	0027F	.BYTE	1
20	20	43	42	4C	42	00280	.ASCII	\BLBC \



					E900	00286	.WORD	-5888	
					05	00288	.BYTE	5	
					20	00289	.BYTE	32	
20	20	53	42	4C	42	0028A	.ASCII	\BLBS \	
					E800	00290	.WORD	-6144	
					05	00292	.BYTE	5	
					20	00293	.BYTE	32	
20	20	51	45	4C	42	00294	.ASCII	\BLEQ \	
					1500	0029A	.WORD	5376	
					04	0029C	.BYTE	4	
					00	0029D	.BYTE	0	
20	55	51	45	4C	42	0029E	.ASCII	\BLEQU \	
					1B00	002A4	.WORD	6912	
					04	002A6	.BYTE	4	
					00	002A7	.BYTE	0	
20	20	53	53	4C	42	002A8	.ASCII	\BLSS \	
					1900	002AE	.WORD	6400	
					04	002B0	.BYTE	4	
					00	002B1	.BYTE	0	
20	55	53	53	4C	42	002B2	.ASCII	\BLSSU \	
					1F00	002B8	.WORD	7936	
					04	002BA	.BYTE	4	
					00	002BB	.BYTE	0	
20	20	51	45	4E	42	002BC	.ASCII	\BNEQ \	
					1200	002C2	.WORD	4608	
					04	002C4	.BYTE	4	
					00	002C5	.BYTE	0	
20	55	51	45	4E	42	002C6	.ASCII	\BNEQU \	
					1200	002CC	.WORD	4608	
					04	002CE	.BYTE	4	
					00	002CF	.BYTE	0	
20	20	20	54	50	42	002D0	.ASCII	\BPT \	
					0300	002D6	.WORD	768	
					00	002D8	.BYTE	0	
					00	002D9	.BYTE	0	
20	20	20	42	52	42	002DA	.ASCII	\BRB \	
					1100	002E0	.WORD	4352	
					04	002E2	.BYTE	4	
					00	002E3	.BYTE	0	
20	20	20	57	52	42	002E4	.ASCII	\BRW \	
					3100	002EA	.WORD	12544	
					04	002EC	.BYTE	4	
					01	002ED	.BYTE	1	
20	20	42	42	53	42	002EE	.ASCII	\BSBB \	
					1000	002F4	.WORD	4096	
					04	002F6	.BYTE	4	
					00	002F7	.BYTE	0	
20	20	57	42	53	42	002F8	.ASCII	\BSBW \	
					3000	002FE	.WORD	12288	
					04	00300	.BYTE	4	
					01	00301	.BYTE	1	
20	20	4C	47	55	42	00302	.ASCII	\BUGL \	
					FDF	00308	.WORD	-513	
					00	0030A	.BYTE	0	
					00	0030B	.BYTE	0	
20	20	57	47	55	42	0030C	.ASCII	\BUGW \	
					FEFF	00312	.WORD	-257	

					00	00314	.BYTE	0	
					00	00315	.BYTE	0	
20	20	20	43	56	42	00316	.ASCII	\BVC	\
					1C00	0031C	.WORD	7168	
					04	0031E	.BYTE	4	
					00	0031F	.BYTE	0	
20	20	20	53	56	42	00320	.ASCII	\BVS	\
					1D00	00326	.WORD	7424	
					04	00328	.BYTE	4	
					00	00329	.BYTE	0	
20	47	4C	4C	41	43	0032A	.ASCII	\CALLG	\
					FA00	00330	.WORD	-1536	
					08	00332	.BYTE	8	
					00	00333	.BYTE	0	
20	53	4C	4C	41	43	00334	.ASCII	\CALLS	\
					FB00	0033A	.WORD	-1280	
					08	0033C	.BYTE	8	
					20	0033D	.BYTE	32	
20	42	45	53	41	43	0033E	.ASCII	\CASEB	\
					8F00	00344	.WORD	-28928	
					11	00346	.BYTE	17	
					00	00347	.BYTE	0	
20	4C	45	53	41	43	00348	.ASCII	\CASEL	\
					CF00	0034E	.WORD	-12544	
					11	00350	.BYTE	17	
					02	00351	.BYTE	2	
20	57	45	53	41	43	00352	.ASCII	\CASEW	\
					AF00	00358	.WORD	-20736	
					11	0035A	.BYTE	17	
					01	0035B	.BYTE	1	
20	20	45	4D	48	43	0035C	.ASCII	\CHME	\
					BD00	00362	.WORD	-17152	
					01	00364	.BYTE	1	
					01	00365	.BYTE	1	
20	20	4B	4D	48	43	00366	.ASCII	\CHMK	\
					BC00	0036C	.WORD	-17408	
					01	0036E	.BYTE	1	
					01	0036F	.BYTE	1	
20	20	53	4D	48	43	00370	.ASCII	\CHMS	\
					BE00	00376	.WORD	-16896	
					01	00378	.BYTE	1	
					01	00379	.BYTE	1	
20	20	55	4D	48	43	0037A	.ASCII	\CHMU	\
					BF00	00380	.WORD	-16640	
					01	00382	.BYTE	1	
					01	00383	.BYTE	1	
20	20	42	52	4C	43	00384	.ASCII	\CLRB	\
					9400	0038A	.WORD	-27648	
					01	0038C	.BYTE	1	
					00	0038D	.BYTE	0	
20	20	44	52	4C	43	0038E	.ASCII	\CLRD	\
					7C00	00394	.WORD	31744	
					01	00396	.BYTE	1	
					06	00397	.BYTE	6	
20	20	46	52	4C	43	00398	.ASCII	\CLRF	\
					D400	0039E	.WORD	-11264	
					01	003A0	.BYTE	1	

.....

				05	003A1	.BYTE	5	
20	20	47	52	4C	003A2	.ASCII	\CLRG	\
				7C00	003A8	.WORD	31744	
				01	003AA	.BYTE	1	
				07	003AB	.BYTE	7	
20	20	48	52	4C	003AC	.ASCII	\CLR H	\
				7CFD	003B2	.WORD	31997	
				01	003B4	.BYTE	1	
				08	003B5	.BYTE	8	
20	20	4C	52	4C	003B6	.ASCII	\CLRL	\
				D400	003BC	.WORD	-11264	
				01	003BE	.BYTE	1	
				02	003BF	.BYTE	2	
20	20	4F	52	4C	003C0	.ASCII	\CLRO	\
				7CFD	003C6	.WORD	31997	
				01	003C8	.BYTE	1	
				04	003C9	.BYTE	4	
20	20	51	52	4C	003CA	.ASCII	\CLRO	\
				7C00	003D0	.WORD	31744	
				01	003D2	.BYTE	1	
				03	003D3	.BYTE	3	
20	20	57	52	4C	003D4	.ASCII	\CLRW	\
				B400	003DA	.WORD	-19456	
				01	003DC	.BYTE	1	
				01	003DD	.BYTE	1	
20	20	42	50	4D	003DE	.ASCII	\CMPB	\
				9100	003E4	.WORD	-28416	
				02	003E6	.BYTE	2	
				00	003E7	.BYTE	0	
20	33	43	50	4D	003E8	.ASCII	\CMPC3	\
				2900	003EE	.WORD	10496	
				0A	003F0	.BYTE	10	
				B0	003F1	.BYTE	-80	
20	35	43	50	4D	003F2	.ASCII	\CMPC5	\
				2D00	003F8	.WORD	11520	
				0C	003FA	.BYTE	12	
				BB	003FB	.BYTE	-69	
20	20	44	50	4D	003FC	.ASCII	\CMPD	\
				7100	00402	.WORD	28928	
				02	00404	.BYTE	2	
				06	00405	.BYTE	6	
20	20	46	50	4D	00406	.ASCII	\CMPF	\
				5100	0040C	.WORD	20736	
				02	0040E	.BYTE	2	
				05	0040F	.BYTE	5	
20	20	47	50	4D	00410	.ASCII	\CMPG	\
				51FD	00416	.WORD	20989	
				02	00418	.BYTE	2	
				07	00419	.BYTE	7	
20	20	48	50	4D	0041A	.ASCII	\CMPH	\
				71FD	00420	.WORD	29181	
				02	00422	.BYTE	2	
				08	00423	.BYTE	8	
20	20	4C	50	4D	00424	.ASCII	\CMPL	\
				D100	0042A	.WORD	-12032	
				02	0042C	.BYTE	2	
				02	0042D	.BYTE	2	

.....



20	33	50	50	4D	43	0042E	.ASCII	\CMPP3 \
					3500	00434	.WORD	13568
					0A	00436	.BYTE	10
					C0	00437	.BYTE	-64
20	34	50	50	4D	43	00438	.ASCII	\CMPP4 \
					3700	0043E	.WORD	14080
					02	00440	.BYTE	2
					0C	00441	.BYTE	12
20	20	56	50	4D	43	00442	.ASCII	\CMPV \
					EC00	00448	.WORD	-5120
					08	0044A	.BYTE	8
					D2	0044B	.BYTE	-46
20	20	57	50	4D	43	0044C	.ASCII	\CMPW \
					B100	00452	.WORD	-20224
					02	00454	.BYTE	2
					01	00455	.BYTE	1
20	56	5A	50	4D	43	00456	.ASCII	\CMPZV \
					ED00	0045C	.WORD	-4864
					08	0045E	.BYTE	8
					D2	0045F	.BYTE	-46
20	20	20	43	52	43	00460	.ASCII	\CRC \
					0B00	00466	.WORD	2816
					16	00468	.BYTE	22
					00	00469	.BYTE	0
20	44	42	54	56	43	0046A	.ASCII	\CVTBD \
					6C00	00470	.WORD	27648
					08	00472	.BYTE	8
					06	00473	.BYTE	6
20	46	42	54	56	43	00474	.ASCII	\CVTBF \
					4C00	0047A	.WORD	19456
					08	0047C	.BYTE	8
					05	0047D	.BYTE	5
20	47	42	54	56	43	0047E	.ASCII	\CVTBG \
					4CFD	00484	.WORD	19709
					08	00486	.BYTE	8
					07	00487	.BYTE	7
20	48	42	54	56	43	00488	.ASCII	\CVTBH \
					6CFD	0048E	.WORD	27901
					08	00490	.BYTE	8
					08	00491	.BYTE	8
20	4C	42	54	56	43	00492	.ASCII	\CVTBL \
					9800	00498	.WORD	-26624
					08	0049A	.BYTE	8
					02	0049B	.BYTE	2
20	57	42	54	56	43	0049C	.ASCII	\CVTBW \
					9900	004A2	.WORD	-26368
					08	004A4	.BYTE	8
					01	004A5	.BYTE	1
20	42	44	54	56	43	004A6	.ASCII	\CVTDB \
					6800	004AC	.WORD	26624
					08	004AE	.BYTE	8
					60	004AF	.BYTE	96
20	46	44	54	56	43	004B0	.ASCII	\CVTDF \
					7600	004B6	.WORD	30208
					08	004B8	.BYTE	8
					65	004B9	.BYTE	101
20	48	44	54	56	43	004BA	.ASCII	\CVTDH \

					32FD	004C0	.WORD	13053
					08	004C2	.BYTE	8
					68	004C3	.BYTE	104
20	4C	44	54	56	43	004C4	.ASCII	\CVTDL \
					6A00	004CA	.WORD	27136
					08	004CC	.BYTE	8
					62	004CD	.BYTE	98
20	57	44	54	56	43	004CE	.ASCII	\CVTDW \
					6900	004D4	.WORD	26880
					08	004D6	.BYTE	8
					61	004D7	.BYTE	97
20	42	46	54	56	43	004D8	.ASCII	\CVTFB \
					4800	004DE	.WORD	18432
					08	004E0	.BYTE	8
					50	004E1	.BYTE	80
20	44	46	54	56	43	004E2	.ASCII	\CVTFD \
					5600	004E8	.WORD	22016
					08	004EA	.BYTE	8
					56	004EB	.BYTE	86
20	47	46	54	56	43	004EC	.ASCII	\CVTFG \
					99FD	004F2	.WORD	-26115
					08	004F4	.BYTE	8
					57	004F5	.BYTE	87
20	48	46	54	56	43	004F6	.ASCII	\CVTFH \
					98FD	004FC	.WORD	-26371
					08	004FE	.BYTE	8
					58	004FF	.BYTE	88
20	4C	46	54	56	43	00500	.ASCII	\CVTFL \
					4A00	00506	.WORD	18944
					08	00508	.BYTE	8
					52	00509	.BYTE	82
20	57	46	54	56	43	0050A	.ASCII	\CVTFW \
					4900	00510	.WORD	18688
					08	00512	.BYTE	8
					51	00513	.BYTE	81
20	42	47	54	56	43	00514	.ASCII	\CVTGB \
					48FD	0051A	.WORD	18685
					08	0051C	.BYTE	8
					70	0051D	.BYTE	112
20	46	47	54	56	43	0051E	.ASCII	\CVTGF \
					33FD	00524	.WORD	13309
					08	00526	.BYTE	8
					75	00527	.BYTE	117
20	48	47	54	56	43	00528	.ASCII	\CVTGH \
					56FD	0052E	.WORD	22269
					08	00530	.BYTE	8
					78	00531	.BYTE	120
20	4C	47	54	56	43	00532	.ASCII	\CVTGL \
					4AFD	00538	.WORD	19197
					08	0053A	.BYTE	8
					72	0053B	.BYTE	114
20	57	47	54	56	43	0053C	.ASCII	\CVTGW \
					49FD	00542	.WORD	18941
					08	00544	.BYTE	8
					71	00545	.BYTE	113
20	42	48	54	56	43	00546	.ASCII	\CVTHB \
					68FD	0054C	.WORD	26877

				08	0054E	.BYTE	8
				80	0054F	.BYTE	-128
20	44	48	54	56	43 00550	.ASCII	\CVTHD \
				F7FD	00556	.WORD	-2051
				08	00558	.BYTE	8
				86	00559	.BYTE	-122
20	46	48	54	56	43 0055A	.ASCII	\CVTHF \
				F6FD	00560	.WORD	-2307
				08	00562	.BYTE	8
				85	00563	.BYTE	-123
20	47	48	54	56	43 00564	.ASCII	\CVTHG \
				76FD	0056A	.WORD	30461
				08	0056C	.BYTE	8
				87	0056D	.BYTE	-121
20	4C	48	54	56	43 0056E	.ASCII	\CVTHL \
				6AFD	00574	.WORD	27389
				08	00576	.BYTE	8
				82	00577	.BYTE	-126
20	57	48	54	56	43 00578	.ASCII	\CVTHW \
				69FD	0057E	.WORD	27133
				08	00580	.BYTE	8
				81	00581	.BYTE	-127
20	42	4C	54	56	43 00582	.ASCII	\CVTLB \
				F600	00588	.WORD	-2560
				08	0058A	.BYTE	8
				20	0058B	.BYTE	32
20	44	4C	54	56	43 0058C	.ASCII	\CVTLD \
				6E00	00592	.WORD	28160
				08	00594	.BYTE	8
				26	00595	.BYTE	38
20	46	4C	54	56	43 00596	.ASCII	\CVTLF \
				4E00	0059C	.WORD	19968
				08	0059E	.BYTE	8
				25	0059F	.BYTE	37
20	47	4C	54	56	43 005A0	.ASCII	\CVTLG \
				4EFD	005A6	.WORD	20221
				08	005A8	.BYTE	8
				27	005A9	.BYTE	39
20	48	4C	54	56	43 005AA	.ASCII	\CVTLH \
				6EFD	005B0	.WORD	28413
				08	005B2	.BYTE	8
				28	005B3	.BYTE	40
20	50	4C	54	56	43 005B4	.ASCII	\CVTLP \
				F900	005BA	.WORD	-1792
				08	005BC	.BYTE	8
				2C	005BD	.BYTE	44
20	57	4C	54	56	43 005BE	.ASCII	\CVTLW \
				F700	005C4	.WORD	-2304
				08	005C6	.BYTE	8
				21	005C7	.BYTE	33
20	4C	50	54	56	43 005C8	.ASCII	\CVTPL \
				3600	005CE	.WORD	13824
				08	005D0	.BYTE	8
				C2	005D1	.BYTE	-62
20	53	50	54	56	43 005D2	.ASCII	\CVTPS \
				0800	005D8	.WORD	2048
				08	005DA	.BYTE	8

.....



20	54	50	54	56	43	005DB	.BYTE	-53
					2400	005DC	.ASCII	\CVTPT \
					0C	005E2	.WORD	9216
					CB	005E4	.BYTE	12
4C	44	52	54	56	43	005E5	.BYTE	-53
					6B00	005E6	.ASCII	\CVTRDL \
					08	005EC	.WORD	27392
					62	005EE	.BYTE	8
4C	46	52	54	56	43	005EF	.BYTE	98
					4B00	005F0	.ASCII	\CVTRFL \
					08	005F6	.WORD	19200
					52	005F8	.BYTE	8
4C	47	52	54	56	43	005F9	.BYTE	82
					4BFD	005FA	.ASCII	\CVTRGL \
					08	00600	.WORD	19453
					72	00602	.BYTE	8
4C	48	52	54	56	43	00603	.BYTE	114
					6BFD	00604	.ASCII	\CVTRHL \
					08	0060A	.WORD	27645
					82	0060C	.BYTE	8
20	50	53	54	56	43	0060D	.BYTE	-126
					0900	0060E	.ASCII	\CVTSP \
					08	00614	.WORD	2304
					BC	00616	.BYTE	8
20	50	54	54	56	43	00617	.BYTE	-68
					2600	00618	.ASCII	\CVTTP \
					0C	0061E	.WORD	9728
					BC	00620	.BYTE	12
20	42	57	54	56	43	00621	.BYTE	-68
					3300	00622	.ASCII	\CVTWB \
					08	00628	.WORD	13056
					10	0062A	.BYTE	8
20	44	57	54	56	43	0062B	.BYTE	16
					6D00	0062C	.ASCII	\CVTWD \
					08	00632	.WORD	27904
					16	00634	.BYTE	8
20	46	57	54	56	43	00635	.BYTE	22
					4D00	00636	.ASCII	\CVTWf \
					08	0063C	.WORD	19712
					15	0063E	.BYTE	8
20	47	57	54	56	43	0063F	.BYTE	21
					4DFD	00640	.ASCII	\CVTWG \
					08	00646	.WORD	19965
					17	00648	.BYTE	8
20	48	57	54	56	43	00649	.BYTE	23
					6DFD	0064A	.ASCII	\CVTWH \
					08	00650	.WORD	28157
					18	00652	.BYTE	8
20	4C	57	54	56	43	00653	.BYTE	24
					3200	00654	.ASCII	\CVTWL \
					08	0065A	.WORD	12800
					12	0065C	.BYTE	8
20	20	42	43	45	44	0065D	.BYTE	18
					9700	0065E	.ASCII	\DECB \
					01	00664	.WORD	-26880
					00	00666	.BYTE	1
					00	00667	.BYTE	0

20	20	4C	43	45	44	00668	.ASCII	\DECL	\
					D700	0066E	.WORD	-10496	
					01	00670	.BYTE	1	
					02	00671	.BYTE	2	
20	20	57	43	45	44	00672	.ASCII	\DECM	\
					B700	00678	.WORD	-18688	
					01	0067A	.BYTE	1	
					01	0067B	.BYTE	1	
20	32	42	56	49	44	0067C	.ASCII	\DIVB2	\
					8600	00682	.WORD	-31232	
					02	00684	.BYTE	2	
					00	00685	.BYTE	0	
20	33	42	56	49	44	00686	.ASCII	\DIVB3	\
					8700	0068C	.WORD	-30976	
					03	0068E	.BYTE	3	
					00	0068F	.BYTE	0	
20	32	44	56	49	44	00690	.ASCII	\DIVD2	\
					6600	00696	.WORD	26112	
					02	00698	.BYTE	2	
					06	00699	.BYTE	6	
20	33	44	56	49	44	0069A	.ASCII	\DIVD3	\
					6700	006A0	.WORD	26368	
					03	006A2	.BYTE	3	
					06	006A3	.BYTE	6	
20	32	46	56	49	44	006A4	.ASCII	\DIVF2	\
					4600	006AA	.WORD	17920	
					02	006AC	.BYTE	2	
					05	006AD	.BYTE	5	
20	33	46	56	49	44	006AE	.ASCII	\DIVF3	\
					4700	006B4	.WORD	18176	
					03	006B6	.BYTE	3	
					05	006B7	.BYTE	5	
20	32	47	56	49	44	006B8	.ASCII	\DIVG2	\
					46FD	006BE	.WORD	18173	
					02	006C0	.BYTE	2	
					07	006C1	.BYTE	7	
20	33	47	56	49	44	006C2	.ASCII	\DIVG3	\
					47FD	006C8	.WORD	18429	
					03	006CA	.BYTE	3	
					07	006CB	.BYTE	7	
20	32	48	56	49	44	006CC	.ASCII	\DIVH2	\
					66FD	006D2	.WORD	26365	
					02	006D4	.BYTE	2	
					08	006D5	.BYTE	8	
20	33	48	56	49	44	006D6	.ASCII	\DIVH3	\
					67FD	006DC	.WORD	26621	
					03	006DE	.BYTE	3	
					08	006DF	.BYTE	8	
20	32	4C	56	49	44	006E0	.ASCII	\DIVL2	\
					C600	006E6	.WORD	-14848	
					02	006E8	.BYTE	2	
					02	006E9	.BYTE	2	
20	33	4C	56	49	44	006EA	.ASCII	\DIVL3	\
					C700	006F0	.WORD	-14592	
					03	006F2	.BYTE	3	
					02	006F3	.BYTE	2	
20	20	50	56	49	44	006F4	.ASCII	\DIVP	\

					2700	006FA	.WORD	9984
					03	006FC	.BYTE	3
					0C	006FD	.BYTE	12
20	32	57	56	49	44	006FE	.ASCII	\DIVW2 \
					A600	00704	.WORD	-23040
					02	00706	.BYTE	2
					01	00707	.BYTE	1
20	33	57	56	49	44	00708	.ASCII	\DIVW3 \
					A700	0070E	.WORD	-22784
					03	00710	.BYTE	3
					01	00711	.BYTE	1
43	50	54	49	44	45	00712	.ASCII	\EDITPC\
					3800	00718	.WORD	14336
					0B	0071A	.BYTE	11
					C0	0071B	.BYTE	-64
20	20	56	49	44	45	0071C	.ASCII	\EDIV \
					7B00	00722	.WORD	31488
					12	00724	.BYTE	18
					00	00725	.BYTE	0
20	44	44	4F	4D	45	00726	.ASCII	\EMODD \
					7400	0072C	.WORD	29696
					13	0072E	.BYTE	19
					96	0072F	.BYTE	-106
20	46	44	4F	4D	45	00730	.ASCII	\EMODF \
					5400	00736	.WORD	21504
					13	00738	.BYTE	19
					95	00739	.BYTE	-107
20	47	44	4F	4D	45	0073A	.ASCII	\EMODG \
					54FD	00740	.WORD	21757
					13	00742	.BYTE	19
					A7	00743	.BYTE	-89
20	48	44	4F	4D	45	00744	.ASCII	\EMODH \
					74FD	0074A	.WORD	29949
					13	0074C	.BYTE	19
					A8	0074D	.BYTE	-88
20	20	4C	55	4D	45	0074E	.ASCII	\EMUL \
					7A00	00754	.WORD	31232
					14	00756	.BYTE	20
					00	00757	.BYTE	0
20	20	56	54	58	45	00758	.ASCII	\EXTV \
					EE00	0075E	.WORD	-4608
					08	00760	.BYTE	8
					D2	00761	.BYTE	-46
20	56	5A	54	58	45	00762	.ASCII	\EXTZV \
					EF00	00768	.WORD	-4352
					08	0076A	.BYTE	8
					D2	0076B	.BYTE	-46
20	20	20	43	46	46	0076C	.ASCII	\FFC \
					EB00	00772	.WORD	-5376
					08	00774	.BYTE	8
					D2	00775	.BYTE	-46
20	20	20	53	46	46	00776	.ASCII	\FFS \
					EA00	0077C	.WORD	-5632
					08	0077E	.BYTE	8
					D2	0077F	.BYTE	-46
20	20	54	4C	41	48	00780	.ASCII	\HALT \
					0000	00786	.WORD	0



					00	00788	.BYTE	0	
					00	00789	.BYTE	0	
20	20	42	43	4E	49	0078A	.ASCII	\INCB \	
					9600	00790	.WORD	-27136	
					01	00792	.BYTE	1	
					00	00793	.BYTE	0	
20	20	4C	43	4E	49	00794	.ASCII	\INCL \	
					D600	0079A	.WORD	-10752	
					01	0079C	.BYTE	1	
					02	0079D	.BYTE	2	
20	20	57	43	4E	49	0079E	.ASCII	\INCW \	
					B600	007A4	.WORD	-18944	
					01	007A6	.BYTE	1	
					01	007A7	.BYTE	1	
20	58	45	44	4E	49	007A8	.ASCII	\INDEX \	
					0A00	007AE	.WORD	2560	
					15	007B0	.BYTE	21	
					00	007B1	.BYTE	0	
49	48	51	53	4E	49	007B2	.ASCII	\INSQHI\	
					5C00	007B8	.WORD	23552	
					02	007BA	.BYTE	2	
					00	007BB	.BYTE	0	
49	54	51	53	4E	49	007BC	.ASCII	\INSQTI\	
					5D00	007C2	.WORD	23808	
					02	007C4	.BYTE	2	
					00	007C5	.BYTE	0	
45	55	51	53	4E	49	007C6	.ASCII	\INSQUE\	
					0E00	007CC	.WORD	3584	
					02	007CE	.BYTE	2	
					00	007CF	.BYTE	0	
20	20	56	53	4E	49	007D0	.ASCII	\INSV \	
					F000	007D6	.WORD	-4096	
					08	007D8	.BYTE	8	
					2D	007D9	.BYTE	45	
20	20	20	50	4D	4A	007DA	.ASCII	\JMP \	
					1700	007E0	.WORD	5888	
					01	007E2	.BYTE	1	
					00	007E3	.BYTE	0	
20	20	20	42	53	4A	007E4	.ASCII	\JSB \	
					1600	007EA	.WORD	5632	
					01	007EC	.BYTE	1	
					00	007ED	.BYTE	0	
58	54	43	50	44	4C	007EE	.ASCII	\LDPCTX\	
					0600	007F4	.WORD	1536	
					00	007F6	.BYTE	0	
					00	007F7	.BYTE	0	
20	20	43	43	4F	4C	007F8	.ASCII	\LOCC \	
					3A00	007FE	.WORD	14848	
					08	00800	.BYTE	8	
					0B	00801	.BYTE	11	
43	48	43	54	41	4D	00802	.ASCII	\MATCHC\	
					3900	00808	.WORD	14592	
					02	0080A	.BYTE	2	
					0B	0080B	.BYTE	11	
20	42	4D	4F	43	4D	0080C	.ASCII	\MCOMB \	
					9200	00812	.WORD	-28160	
					02	00814	.BYTE	2	

					00	00815	.BYTE	0	
20	4C	4D	4F	43	4D	00816	.ASCII	\MCOML \	
					D200	0081C	.WORD	-11776	
					02	0081E	.BYTE	2	
					02	0081F	.BYTE	2	
20	57	4D	4F	43	4D	00820	.ASCII	\MCOMW \	
					B200	00826	.WORD	-19968	
					02	00828	.BYTE	2	
					01	00829	.BYTE	1	
20	20	52	50	46	4D	0082A	.ASCII	\MFPR \	
					DB00	00830	.WORD	-9472	
					02	00832	.BYTE	2	
					02	00833	.BYTE	2	
20	42	47	45	4E	4D	00834	.ASCII	\MNEGB \	
					8E00	0083A	.WORD	-29184	
					02	0083C	.BYTE	2	
					00	0083D	.BYTE	0	
20	44	47	45	4E	4D	0083E	.ASCII	\MNEGD \	
					7200	00844	.WORD	29184	
					02	00846	.BYTE	2	
					06	00847	.BYTE	6	
20	46	47	45	4E	4D	00848	.ASCII	\MNEGF \	
					5200	0084E	.WORD	20992	
					02	00850	.BYTE	2	
					05	00851	.BYTE	5	
20	47	47	45	4E	4D	00852	.ASCII	\MNEGG \	
					52FD	00858	.WORD	21245	
					02	0085A	.BYTE	2	
					07	0085B	.BYTE	7	
20	48	47	45	4E	4D	0085C	.ASCII	\MNEGH \	
					72FD	00862	.WORD	29437	
					02	00864	.BYTE	2	
					08	00865	.BYTE	8	
20	4C	47	45	4E	4D	00866	.ASCII	\MNEGL \	
					CE00	0086C	.WORD	-12800	
					02	0086E	.BYTE	2	
					02	0086F	.BYTE	2	
20	57	47	45	4E	4D	00870	.ASCII	\MNEGW \	
					AE00	00876	.WORD	-20992	
					02	00878	.BYTE	2	
					01	00879	.BYTE	1	
20	42	41	56	4F	4D	0087A	.ASCII	\MOVAB \	
					9E00	00880	.WORD	-25088	
					08	00882	.BYTE	8	
					02	00883	.BYTE	2	
20	44	41	56	4F	4D	00884	.ASCII	\MOVAD \	
					7E00	0088A	.WORD	32256	
					08	0088C	.BYTE	8	
					62	0088D	.BYTE	98	
20	46	41	56	4F	4D	0088E	.ASCII	\MOVAF \	
					DE00	00894	.WORD	-8704	
					08	00896	.BYTE	8	
					52	00897	.BYTE	82	
20	47	41	56	4F	4D	00898	.ASCII	\MOVAG \	
					7E00	0089E	.WORD	32256	
					08	008A0	.BYTE	8	
					72	008A1	.BYTE	114	

20	48	41	56	4F	4D	008A2	.ASCII	\MOVAH \
					7EFD	008A8	.WORD	32509
					08	008AA	.BYTE	8
					82	008AB	.BYTE	-126
20	4C	41	56	4F	4D	008AC	.ASCII	\MOVAL \
					DE00	008B2	.WORD	-8704
					08	008B4	.BYTE	8
					22	008B5	.BYTE	34
20	4F	41	56	4F	4D	008B6	.ASCII	\MOVAO \
					7EFD	008BC	.WORD	32509
					08	008BE	.BYTE	8
					42	008BF	.BYTE	66
20	51	41	56	4F	4D	008C0	.ASCII	\MOVAG \
					7E00	008C6	.WORD	32256
					08	008C8	.BYTE	8
					32	008C9	.BYTE	50
20	57	41	56	4F	4D	008CA	.ASCII	\MOVAM \
					3E00	008D0	.WORD	15872
					08	008D2	.BYTE	8
					12	008D3	.BYTE	18
20	20	42	56	4F	4D	008D4	.ASCII	\MOVAB \
					9000	008DA	.WORD	-28672
					02	008DC	.BYTE	2
					00	008DD	.BYTE	0
20	33	43	56	4F	4D	008DE	.ASCII	\MOVCA \
					2800	008E4	.WORD	10240
					0A	008E6	.BYTE	10
					80	008E7	.BYTE	-80
20	35	43	56	4F	4D	008E8	.ASCII	\MOVCC \
					2C00	008EE	.WORD	11264
					0C	008F0	.BYTE	12
					BB	008F1	.BYTE	-69
20	20	44	56	4F	4D	008F2	.ASCII	\MOVDB \
					7000	008F8	.WORD	28672
					02	008FA	.BYTE	2
					06	008FB	.BYTE	6
20	20	46	56	4F	4D	008FC	.ASCII	\MOVFB \
					5000	00902	.WORD	20480
					02	00904	.BYTE	2
					05	00905	.BYTE	5
20	20	47	56	4F	4D	00906	.ASCII	\MOVGB \
					50FD	0090C	.WORD	20733
					02	0090E	.BYTE	2
					07	0090F	.BYTE	7
20	20	48	56	4F	4D	00910	.ASCII	\MOVHB \
					70FD	00916	.WORD	28925
					02	00918	.BYTE	2
					08	00919	.BYTE	8
20	20	4C	56	4F	4D	0091A	.ASCII	\MOVLB \
					D000	00920	.WORD	-12288
					02	00922	.BYTE	2
					02	00923	.BYTE	2
20	20	4F	56	4F	4D	00924	.ASCII	\MOVQB \
					7DFD	0092A	.WORD	32253
					02	0092C	.BYTE	2
					04	0092D	.BYTE	4
20	20	50	56	4F	4D	0092E	.ASCII	\MOVPB \



					3400	00934	.WORD	13312
					0A	00936	.BYTE	10
					C0	00937	.BYTE	-64
4C	53	50	56	4F	4D	00938	.ASCII	\MOVPSL\
					DC00	0093E	.WORD	-9216
					01	00940	.BYTE	1
					02	00941	.BYTE	2
20	20	51	56	4F	4D	00942	.ASCII	\MOVQ \
					7D00	00948	.WORD	32000
					02	0094A	.BYTE	2
					03	0094B	.BYTE	3
20	43	54	56	4F	4D	0094C	.ASCII	\MOVTC \
					2E00	00952	.WORD	11776
					0D	00954	.BYTE	13
					BB	00955	.BYTE	-69
43	55	54	56	4F	4D	00956	.ASCII	\MOVTCU\
					2F00	0095C	.WORD	12032
					0D	0095E	.BYTE	13
					BB	0095F	.BYTE	-69
20	20	57	56	4F	4D	00960	.ASCII	\MOVW \
					B000	00966	.WORD	-20480
					02	00968	.BYTE	2
					01	00969	.BYTE	1
4C	42	5A	56	4F	4D	0096A	.ASCII	\MOVZBL\
					9A00	00970	.WORD	-26112
					08	00972	.BYTE	8
					02	00973	.BYTE	2
57	42	5A	56	4F	4D	00974	.ASCII	\MOVZBW\
					9B00	0097A	.WORD	-25856
					08	0097C	.BYTE	8
					01	0097D	.BYTE	1
4C	57	5A	56	4F	4D	0097E	.ASCII	\MOVZWL\
					3C00	00984	.WORD	15360
					08	00986	.BYTE	8
					12	00987	.BYTE	18
20	20	52	50	54	4D	00988	.ASCII	\MTPR \
					DA00	0098E	.WORD	-9728
					02	00990	.BYTE	2
					02	00991	.BYTE	2
20	32	42	4C	55	4D	00992	.ASCII	\MULB2 \
					8400	00998	.WORD	-31744
					02	0099A	.BYTE	2
					00	0099B	.BYTE	0
20	33	42	4C	55	4D	0099C	.ASCII	\MULB3 \
					8500	009A2	.WORD	-31488
					03	009A4	.BYTE	3
					00	009A5	.BYTE	0
20	32	44	4C	55	4D	009A6	.ASCII	\MULD2 \
					6400	009AC	.WORD	25600
					02	009AE	.BYTE	2
					06	009AF	.BYTE	6
20	33	44	4C	55	4D	009B0	.ASCII	\MULD3 \
					6500	009B6	.WORD	25856
					03	009B8	.BYTE	3
					06	009B9	.BYTE	6
20	32	46	4C	55	4D	009BA	.ASCII	\MULF2 \
					4400	009C0	.WORD	17408

				02	009C2	.BYTE	2	
				05	009C3	.BYTE	3	
20	33	46	4C	55	4D 009C4	.ASCII	\MULF3 \	
				4500	009CA	.WORD	17664	
				03	009CC	.BYTE	3	
				05	009CD	.BYTE	5	
20	32	47	4C	55	4D 009CE	.ASCII	\MULG2 \	
				44FD	009D4	.WORD	17661	
				02	009D6	.BYTE	2	
				07	009D7	.BYTE	7	
20	33	47	4C	55	4D 009D8	.ASCII	\MULG3 \	
				45FD	009DE	.WORD	17917	
				03	009E0	.BYTE	3	
				07	009E1	.BYTE	7	
20	32	48	4C	55	4D 009E2	.ASCII	\MULH2 \	
				64FD	009E8	.WORD	25853	
				02	009EA	.BYTE	2	
				08	009EB	.BYTE	8	
20	33	48	4C	55	4D 009EC	.ASCII	\MULH3 \	
				65FD	009F2	.WORD	26109	
				03	009F4	.BYTE	3	
				08	009F5	.BYTE	8	
20	32	4C	4C	55	4D 009F6	.ASCII	\MULL2 \	
				C400	009FC	.WORD	-15360	
				02	009FE	.BYTE	2	
				02	009FF	.BYTE	2	
20	33	4C	4C	55	4D 00A00	.ASCII	\MULL3 \	
				C500	00A06	.WORD	-15104	
				03	00A08	.BYTE	3	
				02	00A09	.BYTE	2	
20	20	50	4C	55	4D 00A0A	.ASCII	\MULP \	
				2500	00A10	.WORD	9472	
				03	00A12	.BYTE	3	
				0C	00A13	.BYTE	12	
20	32	57	4C	55	4D 00A14	.ASCII	\MULW2 \	
				A400	00A1A	.WORD	-23552	
				02	00A1C	.BYTE	2	
				01	00A1D	.BYTE	1	
20	33	57	4C	55	4D 00A1E	.ASCII	\MULW3 \	
				A500	00A24	.WORD	-23296	
				03	00A26	.BYTE	3	
				01	00A27	.BYTE	1	
20	20	20	50	4F	4E 00A28	.ASCII	\NOP \	
				0100	00A2E	.WORD	256	
				00	00A30	.BYTE	0	
				00	00A31	.BYTE	0	
20	44	59	4C	4F	50 00A32	.ASCII	\POLYD \	
				7500	00A38	.WORD	29952	
				08	00A3A	.BYTE	8	
				6B	00A3B	.BYTE	107	
20	46	59	4C	4F	50 00A3C	.ASCII	\POLYF \	
				5500	00A42	.WORD	21760	
				08	00A44	.BYTE	8	
				5B	00A45	.BYTE	91	
20	47	59	4C	4F	50 00A46	.ASCII	\POLYG \	
				55FD	00A4C	.WORD	22013	
				08	00A4E	.BYTE	8	

				7B	00A4F	.BYTE	123
20	48	59	4C	4F	50 00A50	.ASCII	\POLYH \
				75FD	00A56	.WORD	30205
				08	00A58	.BYTE	8
				8B	00A59	.BYTE	-117
20	20	52	50	4F	50 00A5A	.ASCII	\POPR \
				BA00	00A60	.WORD	-17920
				01	00A62	.BYTE	1
				01	00A63	.BYTE	1
52	45	42	4F	52	50 00A64	.ASCII	\PROBER\
				0C00	00A6A	.WORD	3072
				08	00A6C	.BYTE	8
				0B	00A6D	.BYTE	11
57	45	42	4F	52	50 00A6E	.ASCII	\PROBEW\
				0D00	00A74	.WORD	3328
				08	00A76	.BYTE	8
				0B	00A77	.BYTE	11
42	41	48	53	55	50 00A78	.ASCII	\PUSHAB\
				9F00	00A7E	.WORD	-24832
				01	00A80	.BYTE	1
				00	00A81	.BYTE	0
44	41	48	53	55	50 00A82	.ASCII	\PUSHAD\
				7F00	00A88	.WORD	32512
				01	00A8A	.BYTE	1
				06	00A8B	.BYTE	6
46	41	48	53	55	50 00A8C	.ASCII	\PUSHAF\
				DF00	00A92	.WORD	-8448
				01	00A94	.BYTE	1
				05	00A95	.BYTE	5
47	41	48	53	55	50 00A96	.ASCII	\PUSHAG\
				7F00	00A9C	.WORD	32512
				01	00A9E	.BYTE	1
				07	00A9F	.BYTE	7
48	41	48	53	55	50 00AA0	.ASCII	\PUSHAH\
				7FFD	00AA6	.WORD	32765
				01	00AA8	.BYTE	1
				08	00AA9	.BYTE	8
4C	41	48	53	55	50 00AAA	.ASCII	\PUSHAL\
				DF00	00AB0	.WORD	-8448
				01	00AB2	.BYTE	1
				02	00AB3	.BYTE	2
4F	41	48	53	55	50 00AB4	.ASCII	\PUSHAO\
				7FFD	00ABA	.WORD	32765
				01	00ABC	.BYTE	1
				04	00ABD	.BYTE	4
51	41	48	53	55	50 00ABE	.ASCII	\PUSHAQ\
				7F00	00AC4	.WORD	32512
				01	00AC6	.BYTE	1
				03	00AC7	.BYTE	3
57	41	48	53	55	50 00AC8	.ASCII	\PUSHAW\
				3F00	00ACE	.WORD	16128
				01	00AD0	.BYTE	1
				01	00AD1	.BYTE	1
20	4C	48	53	55	50 00AD2	.ASCII	\PUSHL \
				DD00	00ADB	.WORD	-8960
				01	00ADA	.BYTE	1
				02	00ADB	.BYTE	2



20	52	48	53	55	50	00ADC	.ASCII	\PUSHR \
					BB00	00AE2	.WORD	-17664
					01	00AE4	.BYTE	1
					01	00AE5	.BYTE	1
20	20	20	49	45	52	00AE6	.ASCII	\REI \
					0200	00AEC	.WORD	512
					00	00AEE	.BYTE	0
					00	00AEF	.BYTE	0
49	48	51	4D	45	52	00AFO	.ASCII	\REMQHI\
					5E00	00AF6	.WORD	24064
					02	00AF8	.BYTE	2
					00	00AF9	.BYTE	0
49	54	51	4D	45	52	00AFA	.ASCII	\REMQTI\
					5F00	00B00	.WORD	24320
					02	00B02	.BYTE	2
					00	00B03	.BYTE	0
45	55	51	4D	45	52	00B04	.ASCII	\REMQUE\
					0F00	00B0A	.WORD	3840
					02	00B0C	.BYTE	2
					00	00B0D	.BYTE	0
20	20	20	54	45	52	00B0E	.ASCII	\RET \
					0400	00B14	.WORD	1024
					00	00B16	.BYTE	0
					00	00B17	.BYTE	0
20	20	4C	54	4F	52	00B18	.ASCII	\ROTL \
					9C00	00B1E	.WORD	-25600
					10	00B20	.BYTE	16
					02	00B21	.BYTE	2
20	20	20	42	53	52	00B22	.ASCII	\RSB \
					0500	00B28	.WORD	1280
					00	00B2A	.BYTE	0
					00	00B2B	.BYTE	0
20	20	43	57	42	53	00B2C	.ASCII	\SBWC \
					D900	00B32	.WORD	-9984
					02	00B34	.BYTE	2
					02	00B35	.BYTE	2
20	43	4E	41	43	53	00B36	.ASCII	\SCANC \
					2A00	00B3C	.WORD	10752
					0B	00B3E	.BYTE	11
					80	00B3F	.BYTE	-80
20	20	43	50	4B	53	00B40	.ASCII	\SKPC \
					3B00	00B46	.WORD	15104
					0B	00B48	.BYTE	8
					0B	00B49	.BYTE	11
51	45	47	42	4F	53	00B4A	.ASCII	\SOBGEQ\
					F400	00B50	.WORD	-3072
					05	00B52	.BYTE	5
					20	00B53	.BYTE	32
52	54	47	42	4F	53	00B54	.ASCII	\SOBGTR\
					F500	00B5A	.WORD	-2816
					05	00B5C	.BYTE	5
					20	00B5D	.BYTE	32
20	43	4E	41	50	53	00B5E	.ASCII	\SPANC \
					2B00	00B64	.WORD	11008
					0B	00B66	.BYTE	11
					80	00B67	.BYTE	-80
20	32	42	42	55	53	00B68	.ASCII	\SUBB2 \

				8200	00B6E	.WORD	-32256
				02	00B70	.BYTE	2
				00	00B71	.BYTE	0
20	33	42	42	55	00B72	.ASCII	\SUBB3 \
				8300	00B78	.WORD	-32000
				03	00B7A	.BYTE	0
				00	00B7B	.BYTE	0
20	32	44	42	55	00B7C	.ASCII	\SUBD2 \
				6200	00B82	.WORD	25088
				02	00B84	.BYTE	2
				06	00B85	.BYTE	6
20	33	44	42	55	00B86	.ASCII	\SUBD3 \
				6300	00B8C	.WORD	25344
				03	00B8E	.BYTE	3
				06	00B8F	.BYTE	6
20	32	46	42	55	00B90	.ASCII	\SUBF2 \
				4200	00B96	.WORD	16896
				02	00B98	.BYTE	2
				03	00B99	.BYTE	3
20	33	46	42	55	00B9A	.ASCII	\SUBF3 \
				4300	00BA0	.WORD	17152
				03	00BA2	.BYTE	3
				05	00BA3	.BYTE	5
20	32	47	42	55	00BA4	.ASCII	\SUBG2 \
				42FD	00BAA	.WORD	17149
				02	00BAC	.BYTE	2
				07	00BAD	.BYTE	7
20	33	47	42	55	00BAE	.ASCII	\SUBG3 \
				43FD	00BB4	.WORD	17405
				03	00BB6	.BYTE	3
				07	00BB7	.BYTE	7
20	32	48	42	55	00BB8	.ASCII	\SUBH2 \
				62FD	00BBE	.WORD	25341
				02	00BC0	.BYTE	2
				08	00BC1	.BYTE	8
20	33	48	42	55	00BC2	.ASCII	\SUBH3 \
				63FD	00BC8	.WORD	25597
				03	00BCA	.BYTE	3
				08	00BCB	.BYTE	8
20	32	4C	42	55	00BCC	.ASCII	\SUBL2 \
				C200	00BD2	.WORD	-15872
				02	00BD4	.BYTE	2
				02	00BD5	.BYTE	2
20	33	4C	42	55	00BD6	.ASCII	\SUBL3 \
				C300	00BDC	.WORD	-15616
				03	00BDE	.BYTE	3
				02	00BDF	.BYTE	2
20	34	50	42	55	00BE0	.ASCII	\SUBP4 \
				2200	00BE6	.WORD	8704
				02	00BE8	.BYTE	2
				0C	00BE9	.BYTE	12
20	36	50	42	55	00BEA	.ASCII	\SUBP6 \
				2300	00BFD	.WORD	8960
				03	00BF2	.BYTE	3
				0C	00BF3	.BYTE	12
20	32	57	42	55	00BF4	.ASCII	\SUBW2 \
				A200	00BFA	.WORD	-24064

.....

				02	00BFC	.BYTE	2	
				01	00BFD	.BYTE	1	
20	33	57	42	55	53	.ASCII	1	\SUBW3 \
				A300	00C04	.WORD	-23808	
				03	00C06	.BYTE	3	
				01	00C07	.BYTE	1	
58	54	43	50	56	53	.ASCII	1	\SVPCTX\
				0700	00C0E	.WORD	1792	
				00	00C10	.BYTE	0	
				00	00C11	.BYTE	0	
20	20	42	54	53	54	.ASCII	0	\TSTB \
				9500	00C18	.WORD	-27392	
				01	00C1A	.BYTE	1	
				00	00C1B	.BYTE	0	
20	20	44	54	53	54	.ASCII	0	\TSTD \
				7300	00C22	.WORD	29440	
				01	00C24	.BYTE	1	
				06	00C25	.BYTE	6	
20	20	46	54	53	54	.ASCII	0	\TSTF \
				5300	00C2C	.WORD	21248	
				01	00C2E	.BYTE	1	
				05	00C2F	.BYTE	5	
20	20	47	54	53	54	.ASCII	0	\TSTG \
				53FD	00C36	.WORD	21501	
				01	00C38	.BYTE	1	
				07	00C39	.BYTE	7	
20	20	48	54	53	54	.ASCII	0	\TSTH \
				73FD	00C40	.WORD	29693	
				01	00C42	.BYTE	1	
				08	00C43	.BYTE	8	
20	20	4C	54	53	54	.ASCII	0	\TSTL \
				D500	00C4A	.WORD	-11008	
				01	00C4C	.BYTE	1	
				02	00C4D	.BYTE	2	
20	20	57	54	53	54	.ASCII	0	\TSTW \
				B500	00C54	.WORD	-19200	
				01	00C56	.BYTE	1	
				01	00C57	.BYTE	1	
20	20	20	43	46	58	.ASCII	0	\XFC \
				FC00	00C5E	.WORD	-1024	
				00	00C60	.BYTE	0	
				00	00C61	.BYTE	0	
20	32	42	52	4F	58	.ASCII	0	\XORB2 \
				8C00	00C68	.WORD	-29696	
				02	00C6A	.BYTE	2	
				00	00C6B	.BYTE	0	
20	33	42	52	4F	58	.ASCII	0	\XORB3 \
				8D00	00C72	.WORD	-29440	
				03	00C74	.BYTE	3	
				00	00C75	.BYTE	0	
20	32	4C	52	4F	58	.ASCII	0	\XORL2 \
				CC00	00C7C	.WORD	-13312	
				02	00C7E	.BYTE	2	
				02	00C7F	.BYTE	2	
20	33	4C	52	4F	58	.ASCII	0	\XORL3 \
				CD00	00C86	.WORD	-13056	
				03	00C88	.BYTE	3	



```

        .BYTE
        .ASCII
        .WORD
        .BYTE
        .BYTE
        .ASCII
        .WORD
        .BYTE
        .BYTE
        .BLKB
DBG$OPCODE_KIND
        .WORD

        .BYTE
        .WORD
        .BYTE
        .WORD

        .BYTE
        .WORD

        .BYTE
        .WORD

        .BYTE
        .WORD
        .BYTE
        .WORD
        .WORD

        .BYTE
        .WORD

```

```

.BYTE 2
.ASCII \XORW2 \
.WORD -21504
.BYTE 2
.BYTE 1
.ASCII \XORW3 \
.WORD -21248
.BYTE 3
.BYTE 1
.BLKB 2
ODE KIND -TABLE::
.WORD -192, 260, 279, 72, 283, 285, 203, 308, -
149, 155, 196, 112, 266, 267, 199, 282, -
75, 73, 70, 41, 45, 66, 202, 201, 43, 68, -
46, 67, 79, 80, 44, 69, 21, 22, 304, 305, -
150, 257, 156, 178, 227, 100, 287, 291, -
228, 101, 238, 239, 76, 74, 162, 157, -
235, 107, 148, 108, 181, 205, 204, 288, -
243, 7, 225, 276, 13, 14, 296, 297, 246, -
250, 170, 171, 124, 129, 128, 152, 114, -
159, 143, 3, 230, 103, 212, 311, 184, -
262, 125
.BYTE 0[2]
.WORD 8
.BYTE 0[6]
.WORD 197, 198, 280, 281, 11, 12, 294, 295, -
247, 248, 168, 169, 119, 123, 122, 151, -
113, 158, 142, 2, 229, 102, 211, 310, -
183, 261, 120
.BYTE 0[2]
.WORD 28, 30, 187, 182, 97, 237, 224, 275, 9, -
10, 292, 293, 245, 246, 166, 167, 54, 55, -
47, 48, 317, 318, 210, 83, 226, 99, 206, -
61, 90, 309, 193, 163, 117, 118, 241, -
242, 284, 1, 217, 268, 23, 24, 306, 307, -
258, 259, 179, 180, 59, 60, 52, 53, 321, -
322, 216, 85, 240, 110, 208, 63, 98, 315, -
195, 165, 58, 51, 265, 278, 87, 86, 88, -
89, 19, 20, 302, 303, 255, 256, 176, 177, -
56, 57, 49, 50, 319, 320, 215, 84, 233, -
106, 207, 62, 95, 314, 194, 164, 25, 286, -
244, 209, 236, 277, 222, 273, 35, 31, 37, -
34, 36, 32, 38, 33, 65, 64, 191, 190, -
109, 111, 188, 189, 200, 6, 27, 26, 289, -
290, 141, 147, 29, 146, 81, 82, 316
.BYTE 0[6]
.WORD 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -
0, 0
.BYTE 0[68]
.WORD 121, 131
.BYTE 0[24]
.WORD 15, 16, 298, 299, 251, 252, 172, 173, -
130, 134, 133, 153, 115, 160, 144, 4, -
231, 104, 213, 312, 185, 263, 132
.BYTE 0[18]
.WORD 17, 18, 300, 301, 253, 254, 174, 175, -
135, 140, 139, 154, 116, 161, 145, 5, -

```



```

02 00 010EB .BYTE 0 2
    46 010ED .ASCII \f\
    50 010EE .ASCII \p\
02 00 010EF .BYTE 0 2
    53 010F1 .ASCII \s\
    50 010F2 .ASCII \p\
02 00 010F3 .BYTE 0 2
    50 010F5 .ASCII \p\
    43 010F6 .ASCII \c\
02 00 010F7 .BYTE 0 2
    3F 010F9 .ASCII \? \
    3F 010FA .ASCII \? \
02 00 010FB .BYTE 0 2
    3F 010FD .ASCII \? \
    3F 010FE .ASCII \? \
02 00 010FF .BYTE 0 2
    49 01101 .ASCII \i\
    56 01102 .ASCII \v\
02 00 01103 .BYTE 0 2
    44 01105 .ASCII \d\
    56 01106 .ASCII \v\
    00 01107 .BYTE 0
44 41 21 03 01108 P.AAC: .ASCII <3>\!AD\
43 41 21 03 0110C P.AAD: .ASCII <3>\!AC\
    2C 01110 P.AAE: .ASCII \, \

```

.PSECT DBG\$OWN,NOEXE, PIC,2

00000 OP\_BUFFER:  
.BLKB 16

DBG\$OPCODE\_NAME\_TABLE==

OPERAND\_VALUE= P.AAA  
REGISTER\_NAME= OP\_BUFFER  
FORMAT\_AD= P.AAB  
FORMAT\_AC= P.AAC  
COMMA= P.AAD  
P.AAE

```

.EXTRN DBG$GB_RADIX,DBG$CONV_TEXT_VALUE
.EXTRN DBG$COVER_DX,DX
.EXTRN DBG$PRINT,DBG$PRINT_VALUE
.EXTRN DBG$PRINT_IDENTIFIER_PC
.EXTRN DBG$NEWLINE,DBG$POP_TEMPHEM
.EXTRN DBG$PUSH_TEMPHEM
.EXTRN DBG$IS_IT_ENTRY
.EXTRN DBG$MAKE_VAL_DESC
.EXTRN DBG$NPARSE_ADDRESS
.EXTRN DBG$NPARSE_EXPRESSION
.EXTRN DBG$PRIM_TO_VAL
.EXTRN SYS$UNWIND

```

.PSECT DBG\$CODE,NOVRT, SHR, PIC,0

0000 00000 DECODE\_HANDLER:

```

00000920 50 04 AC D0 00002 .WORD Save nothing
          8F 04 AD D1 00006 MOVL SIG_ARGS, R0
          CMPL 4(R0), #2336

```

0793  
0800  
...

	51	0C	20	13	0000E	BEQL	2\$	
	06	04	AC	D0	00010	MOVL	ENABLE_ARGS, R1	
	50	0918	B1	E9	00014	BLBC	04(R1), 1\$	0801
			8F	3C	00018	MOVZWL	#2328, R0	
				04	0001D	RET		
	50	08	AC	D0	0001E	1\$: MOVL	MECH_ARGS, R0	0803
0C	A0	08	B1	D0	00022	MOVL	08(RT), 12(R0)	
			7E	7C	00027	CLRG	-(SP)	0805
00000000G	00		02	FB	00029	CALLS	#2, SYSSUNWIND	
	50		01	D0	00030	2\$: MOVL	#1, R0	0806
			04	00033	RET			0807

; Routine Size: 52 bytes, Routine Base: DBG\$CODE + 0000

:	685	0808	2	
:	686	0809	2	ENABLE Decode_Handler(Signal_Flag, Error_Value);
:	687	0810	2	
:	688	0811	2	Signal_Flag = .Print_Flag;
:	689	0812	2	Error_Value = .Start_Address + 1;



```

691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727

```

```

Pointer = .Start_Address;

++
Do we have an entry mask
--
IF (IF ActualCount() GTR 2 THEN .Entry_Flag ELSE DBG$Is_It_Entry(.Start_Address))
THEN
    BEGIN
        Error_Value = Error_Value + 1;
        Fetch_Instruction(Pointer,context_wu);
        IF .Print_Flag THEN
            BEGIN
                LOCAL Delimiter,Mask_Bits;

                Mask_Bits = .Operand_Value<0,12,0>+ (.Operand_Value<12,4,0>)*16;

                DBG$Print(UPLIT BYTE (%ASCIC 'entry mask ^M'));

                Delimiter = %C'<';

                IF .Mask_Bits EQL 0 THEN DBG$Print(Format_AD,1,Delimiter) ELSE
                    INCR Index FROM 0 TO 19 DO IF .Mask_Bits<.Index,1,0> THEN
                        BEGIN
                            DBG$Print(Format_AD,1,Delimiter);
                            DBG$Print(Format_AC,Register_Name[.Index]);
                            Delimiter = %C',';
                        END;
                    END;

                DBG$Print(Format_AD,1,UPLIT BYTE('>'));

                IF .Operand_Value<12,2,0> NEQ 0 THEN
                    BEGIN
                        DBG$Newline();
                        SIGNAL(dbg$_entrymask);
                    END;
                END;
            END;
        END;
    END

```

```

729 0850 2
730 0851 2
731 0852 2
732 0853 2
733 0854 2
734 0855 2
735 0856 2
736 0857 2
737 0858 2
738 0859 2
739 0860 2
740 0861 2
741 0862 2
742 0863 2
743 0864 2
744 0865 2
745 0866 2
746 0867 2
747 0868 2
748 0869 2
749 0870 2
750 0871 2
751 0872 2
752 0873 2
753 0874 2
754 0875 2
755 0876 2
756 0877 2
757 0878 2

ELSE ! Not an entry mask - decode actual instruction
BEGIN
LOCAL
Opcode,
Opcode_Index,
Opcode_Entry : REF BLOCK[10,BYTE] FIELD(Opcode_Entry_Fields),
Delimiter,
Start;

Fetch_Instruction(Pointer,context_bu);
Opcode = .Op_Buffer[u_byte];
Opcode_Index = .DBG$Opcode_Kind_Table[.Opcode];
IF (.Opcode_Index EQL 0) THEN
BEGIN
IF (.Opcode GEQU XX'FD') THEN
BEGIN
Fetch_Instruction(Pointer,context_bu);
IF ((.Opcode EQLU XX'FD') AND (.Op_Buffer[u_byte] LSSU XX'FD'))
OR ((.Opcode EQLU XX'FF') AND (.Op_Buffer[u_byte] GEQU XX'FD'))
THEN
Opcode_Index = .DBG$Opcode_Kind_Table[.Op_Buffer[u_byte]+XX'100'];
END;
IF (.Opcode_Index EQL 0) THEN SIGNAL(dbg$_noinstran,1,.Start_Address);
END;

Opcode_Entry = DBG$Opcode_Name_Table[.Opcode_Index,offset];
IF .Print_Flag THEN DBG$Print(Format_AD,6,Opcode_Entry[op_name]);
Delimiter = 'X';

```

```
.. 759      0879      3      State = .Opcode_Entry[op_kind];
.. 760      0880      3
.. 761      0881      3      WHILE (.State NEQ simple_0_operand) DO
.. 762      0882      3      BEGIN
.. 763      0883      3      IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
.. 764      0884      3      Delimiter = '%C',';
.. 765      0885      3
.. 766      0886      3      CASE .State FROM simple_1_operand TO maximum_state OF
.. 767      0887      3      SET
.. 768      0888      3      [simple_1_operand,
.. 769      0889      3      simple_2_operand,
.. 770      0890      3      simple_3_operand]:
.. 771      0891      3      BEGIN
.. 772      0892      3      Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
.. 773      0893      3      State = .State - 1;
.. 774      0894      3      END;
.. 775      0895      3
.. 776      0896      3      [branch_0_operand]:
.. 777      0897      3      BEGIN
.. 778      0898      3      Fetch_Instruction(Pointer,.Opcode_Entry[op_type_one]);
.. 779      0899      3      IF .Print_Flag THEN
.. 780      0900      3      Print_Address(.Operand_Value + .Pointer);
.. 781      0901      3      EXITLOOP;
.. 782      0902      3      END;
.. 783      0903      3
.. 784      0904      3      [branch_1_operand,
.. 785      0905      3      branch_2_operand,
.. 786      0906      3      branch_3_operand]:
.. 787      0907      3      BEGIN
.. 788      0908      3      Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
.. 789      0909      3      State = .State - 1;
.. 790      0910      3      END;
```

```

792 0911 4
793 0912 4 *****
794 0913 4 *****
795 0914 4 *****
796 0915 4 *****
797 0916 4 *****
798 0917 4
799 0918 5
800 0919 5
801 0920 5
802 0921 4
803 0922 4
804 0923 4
805 0924 5
806 0925 5
807 0926 5
808 0927 4
809 0928 4
810 0929 4
811 0930 4
812 0931 5
813 0932 5
814 0933 5
815 0934 6
816 0935 6
817 0936 6
818 0937 5
819 0938 5
820 0939 4
821 0940 4
822 0941 4
823 0942 5
824 0943 5
825 0944 5
826 0945 5
827 0946 4

[convert_datatype,
evaluate_address,
simple_bit_field,
routine_dispatch,
polynomial_value,
probe_for_access,
string_3_operand]:
BEGIN
  Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
  State = Simple_1_Operand;
END;

[string_4_operand]:
BEGIN
  Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
  State = Simple_2_Operand;
END;

[string_5_operand,
string_6_operand]:
BEGIN
  Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
  DECR count FROM .State TO string_5_operand DO
    BEGIN
      IF .Print_Flag THEN DBG$Print(Format AD,1,Delimiter);
      Fetch_Operand(Pointer,context_b,.Print_Flag,0);
    END;
  State = Simple_1_Operand;
END;

[trailing_operand]:
BEGIN
  Fetch_Instruction(Pointer,.Opcode_Entry[op_type_one]);
  IF .Print_Flag THEN Print_Operand(.Opcode_Entry[op_type_one]);
  EXITLOOP;
END;

```



...	829	0947	...
...	830	0948	...
...	831	0949	...
...	832	0950	...
...	833	0951	...
...	834	0952	...
...	835	0953	...
...	836	0954	...
...	837	0955	...
...	838	0956	...
...	839	0957	...
...	840	0958	...
...	841	0959	...
...	842	0960	...
...	843	0961	...
...	844	0962	...
...	845	0963	...
...	846	0964	...
...	847	0965	...
...	848	0966	...
...	849	0967	...
...	850	0968	...
...	851	0969	...
...	852	0970	...
...	853	0971	...
...	854	0972	...
...	855	0973	...
...	856	0974	...
...	857	0975	...
...	858	0976	...
...	859	0977	...
...	860	0978	...
...	861	0979	...
...	862	0980	...
...	863	0981	...
...	864	0982	...
...	865	0983	...
...	866	0984	...
...	867	0985	...
...	868	0986	...
...	869	0987	...
...	870	0988	...
...	871	0989	...
...	872	0990	...
...	873	0991	...
...	874	0992	...
...	875	0993	...

[complex SHIFT]:

```
BEGIN
Fetch_Operand(Pointer,context_b,.Print_Flag,0);
State = Simple_2_Operand;
END;
```

[complex EMOD]:

```
BEGIN
Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
State = Simple_3_Operand;
END;
```

[complex CRC]:

```
BEGIN
Fetch_Operand(Pointer,context_b,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_t,.Print_Flag,0);
EXITLOOP;
END;
```

[complex EMUL]:

```
BEGIN
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_q,.Print_Flag,0);
EXITLOOP;
END;
```

[complex EDIV]:

```
BEGIN
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_g,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
Fetch_Operand(Pointer,context_l,.Print_Flag,0);
EXITLOOP;
END;
```

```

877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933

```

```

[complex INDEX]:
BEGIN
  DECR count FROM 5 TO 0 DO
    BEGIN
      Fetch_Operand(Pointer,context_l,.Print_Flag,0);
      IF .Print_Flag AND (.count NEQ 0) THEN DBGSPrint(Format_AD,1,Delimiter);
    END;
  EXITLOOP;
END;

[complex CASE]:
BEGIN
  LOCAL
    Limit;
    Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
    IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);
    Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
    IF .Print_Flag THEN DBGSPrint(Format_AD,1,Delimiter);

    ++
    Fetch the Limit
    --

    Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);

    ++
    Get the Limit in of the appropriate length
    --
    Limit = (SELECT ONE .Opcode_Entry[ Op_type_one ] OF
      SET
        [context_b]:      .Op_buffer[u_byte];
        [context_w]:      .Op_buffer[u_word];
        [context_l]:      .Op_buffer[u_long];
      TES);

    IF NOT .Print_Flag
    THEN
      Pointer = .Pointer+2*(.Limit+1)
    ELSE
      BEGIN
        LOCAL start;
        start = .Pointer;
        DECR count FROM .Limit TO 0 DO
          BEGIN
            BIND
              CASE Offset = Op_buffer[u_word] : SIGNED;
            $ABORT ON CONTROL Y;
            Fetch_Instruction(Pointer,context_w);
            DBGSNewLine();
            DBGSPrint(Format_AD,2,UPLIT BYTE('
            Print_Address(.CASE_Offset + .Start);
            END;
          END;
        EXITLOOP;
      END;

[complex ASHP]:
BEGIN

```

```

! The number of displacements
!
! What is the operand length?
! Byte
! Word
! Long
!
!
! Convert the word to signed
! Its ok the ^Y out of the ca
!
! Print the address
!
! ASHP didn't fit an

```

```

934 1051 5 Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
935 1052 IF .Print_Flag THEN DBG$Print(Format_AD,1,Delimiter);
936 1053 Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
937 1054 IF .Print_Flag THEN DBG$Print(Format_AD,1,Delimiter);
938 1055 Fetch_Operand(Pointer,.Opcode_Entry[op_type_two],.Print_Flag,0);
939 1056 IF .Print_Flag THEN DBG$Print(Format_AD,1,Delimiter);
940 1057 Fetch_Operand(Pointer,.Opcode_Entry[op_type_one],.Print_Flag,0);
941 1058 EXITLOOP;
942 1059 END;
943 1060
944 1061 [INRANGE,OUTRANGE]:
945 1062 SDBG_ERROR('DBG$Encode_Decode - bad opcode table entry');
946 1063 TES;
947 1064 END;
948 1065 END;
949 1066 RETURN .Pointer;
950 1067 END;
951 1068 1

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
4D 5E 20 6B 73 61 6D 20 79 72 74 6E 65 0D 01111 P.AAF: .ASCII <13>\entry mask ^M\
3E 0111F P.AAG: .ASCII \>\
09 09 01120 P.AAH: .ASCII <9><9>
63 65 44 5F 65 64 6F 63 6E 45 24 47 42 44 2A 01122 P.AAI: .ASCII \*DBG$Encode_Decode - bad opcode table en\
64 6F 63 70 6F 20 64 61 62 20 2D 20 65 64 6F 01131
6E 65 20 65 6C 62 61 74 20 65 01140
79 72 74 0114A .ASCII \try\
CASE_OFFSET= OP_BUFFER
.EXTRN DBG$GV_CONTROL
.PSECT DBG$CODE,NOWRT, SHR, PIC,0
OFFC 00000
.ENTRY DBG$INS_DECODE, Save R2,R3,R4,R5,R6,R7,R8,- 0785
R9,R10,R11
MOVAB LIB$SIGNAL, R11
MOVAB FETCH_OPERAND, R10
MOVAB OPERAND_VALUE, R9
MOVAB DBG$PRINT, R8
MOVAB FORMAT_AD, R7
SUBL2 #20, SP
CLRQ ERROR_VALUE 0786
MOVAL 72$, TFP)
MOVL PRINT_FLAG, R6 0811
MOVL R6, SIGNAL_FLAG
ADDL3 #1, START_ADDRESS, ERROR_VALUE 0812
MOVL START_ADDRESS, POINTER 0813
CMPB (AP), #2 0818
BLEQU 1$
BLBS ENTRY_FLAG, 2$
BRW 8$
PUSHL START_ADDRESS
CALLS #1, DBG$IS_IT_ENTRY
BLBC R0, 8$

```

			0000V	CF	OC	AE	D6	0005A	28:	INCL	ERROR_VALUE	0821
				6A	OC	0A	DD	0005D		PUSHL	#10	0822
50	01	A9		04		AE	9F	0005F		PUSHAB	POINTER	
53		50		50		02	FB	00062		CALLS	#2, FETCH_INSTRUCTION	
		69		0C		56	E9	00067		BLBC	R6, 78	0823
				53		04	EF	0006A		EXTZV	#4, #4, OPERAND_VALUE+1, R0	0827
						10	78	00070		ASHL	#16, R0, R0	
						00	EF	00074		EXTZV	#0, #12, OPERAND_VALUE, MASK_BITS	
						50	CD	00079		ADDL2	R0, MASK_BITS	
					09	A7	9F	0007C		PUSHAB	P.AAF	0829
				68		01	FB	0007F		CALLS	#1, DBGSPRINT	
				6E		3C	DD	00082		MOVL	#60, DELIMITER	0831
						53	DS	00085		TSTL	MASK_BITS	0833
						0B	12	00087		BNEQ	38	
						5E	DD	00089		PUSHL	SP	
						01	DD	0008B		PUSHL	#1	
						57	DD	0008D		PUSHL	R7	
				68		03	FB	0008F		CALLS	#3, DBGSPRINT	
						20	11	00092		BRB	68	
						52	D4	00094	38:	CLRL	INDEX	0834
						52	E1	00096	48:	BBC	INDEX, MASK_BITS, 58	
						5E	DD	0009A		PUSHL	SP	0836
						01	DD	0009C		PUSHL	#1	
						57	DD	0009E		PUSHL	R7	
				68		03	FB	000A0		CALLS	#3, DBGSPRINT	
					B0	A7	DF	000A3		PUSHAL	REGISTER_NAME[INDEX]	0837
					04	A7	9F	000A7		PUSHAB	FORMAT AC	
				68		02	FB	000AA		CALLS	#2, DBGSPRINT	
				6E		2C	DD	000AD		MOVL	#44, DELIMITER	0838
				52		13	F3	000B0	58:	AOBLEQ	#19, INDEX, 48	0834
					17	A7	9F	000B4	68:	PUSHAB	P.AAG	0841
						01	DD	000B7		PUSHL	#1	
						57	DD	000B9		PUSHL	R7	
				68		03	FB	000BB		CALLS	#3, DBGSPRINT	
				30	01	A9	93	000BE		BITB	OPERAND_VALUE+1, #48	0843
						10	13	000C2		BEQL	78	
						00	FB	000C4		CALLS	#0, DBG\$NEWLINE	0845
			000000006	00		8F	DD	000CB		PUSHL	#167843	0846
				68		01	FB	000D1		CALLS	#1, LIB\$SIGNAL	
						03	F2	31	000D4	78:	BRW	718
						09	DD	000D7	88:	PUSHL	#9	0859
					OC	AE	9F	000D9		PUSHAB	POINTER	
						02	FB	000DC		CALLS	#2, FETCH_INSTRUCTION	
						69	9A	000E1		MOVZBL	OP_BUFFER, OPCODE	0860
						52	3C	000E4		MOVZWL	DBG\$OPCODE_KIND_TABLE[OPCODE], OPCODE_INDEX	0861
						4C	12	000EA		BNEQ	128	0862
			000000FD	8F		53	D1	000EC		CMPL	OPCODE, #253	0864
						31	1F	000F3		BLSSU	118	
						09	DD	000F5		PUSHL	#9	0866
					OC	AE	9F	000F7		PUSHAB	POINTER	
						02	FB	000FA		CALLS	#2, FETCH_INSTRUCTION	
			000000FD	8F		53	D1	000FF		CMPL	OPCODE, #253	0867
						06	12	00106		BNEQ	98	
						69	91	00108		CMPB	OP_BUFFER, #253	
						0F	1F	0010C		BLSSU	108	
			000000FF	8F		53	D1	0010E	98:	CMPL	OPCODE, #255	0868
						0F	12	00115		BNEQ	118	



FD	8F	69	91	00117	CMPB	OP BUFFER, #253	
		09	1F	00118	BLSSU	11\$	
	50	69	9A	0011D	MOVZBL	OP BUFFER, R0	0870
	52	FD9B C740	3C	00120	MOVZWL	DBG\$OPCODE_KIND_TABLE+512[R0], OPCODE_INDEX	
		52	D5	00126	TSTL	OPCODE_INDEX	0872
		0E	12	00128	BNEQ	12\$	
		04	AC	DD 0012A	PUSHL	START_ADDRESS	
			01	DD 0012D	PUSHL	#1	
		000281C8	8F	DD 0012F	PUSHL	#164296	
	68		03	FB 00135	CALLS	#3, LIB\$SIGNAL	
	52		0A	C4 00138	MULL2	#10, R2	0875
	54	EEFB C742	9E	0013B	MOVAB	DBG\$OPCODE_NAME_TABLE[R2], OPCODE_ENTRY	
	09		56	E9 00141	BLBC	R6, 13\$	0877
			54	DD 00144	PUSHL	OPCODE_ENTRY	
			06	DD 00146	PUSHL	#6	
			57	DD 00148	PUSHL	R7	
	68		03	FB 0014A	CALLS	#3, DBG\$PRINT	
04	AE		09	D0 0014D	MOVL	#9, DELIMITER	0878
	52	08	A4	9A 00151	MOVZBL	8(OPCODE_ENTRY), STATE	0879
			52	D5 00155	TSTL	STATE	0881
			7A	13 00157	BEQL	20\$	
	0A		56	E9 00159	BLBC	R6, 15\$	0883
		04	AE	9F 0015C	PUSHAB	DELIMITER	
			01	DD 0015F	PUSHL	#1	
			57	DD 00161	PUSHL	R7	
	68		03	FB 00163	CALLS	#3, DBG\$PRINT	
04	AE		2C	D0 00166	MOVL	#44, DELIMITER	0884
	01		52	CF 0016A	CASEL	STATE, #1, #22	0886
					.WORD	18\$-16\$,-	
004A						18\$-16\$,-	
007C	16					18\$-16\$,-	
009A	003E	003E	003E	0016E		18\$-16\$,-	
00F3	0068	0068	0068	00176		19\$-16\$,-	
0171	008E	007C	00D4	0017E		21\$-16\$,-	
	002E	002E	009A	00186		21\$-16\$,-	
	0104	01C4	023D	0018E		21\$-16\$,-	
	02F4	0137	0218	00196		21\$-16\$,-	
						23\$-16\$,-	
						31\$-16\$,-	
						23\$-16\$,-	
						24\$-16\$,-	
						25\$-16\$,-	
						25\$-16\$,-	
						17\$-16\$,-	
						17\$-16\$,-	
						32\$-16\$,-	
						53\$-16\$,-	
						45\$-16\$,-	
						34\$-16\$,-	
						40\$-16\$,-	
						50\$-16\$,-	
						37\$-16\$,-	
						66\$-16\$	
		1A	A7	9F 0019C	PUSHAB	P,AAI	1062
			01	DD 0019F	PUSHL	#1	
		00028362	8F	DD 001A1	PUSHL	#164706	
	68		03	FB 001A7	CALLS	#3, LIB\$SIGNAL	
			A9	11 001AA	BRB	14\$	
			7E	D4 001AC	CLRL	-(SP)	0892

7E	09	A4	04	56	DD	001AE	PUSHL	R6			
				00	EF	001B0	EXTZV	#0	#4, 9(OPCODE_ENTRY), -(SP)		
7E	09	A4	04	28	11	001B6	BRB	22\$			
				00	EF	001B8	EXTZV	#0	#4, 9(OPCODE_ENTRY), -(SP)	0898	
		0000V	CF	0C	9F	001BE	PUSHAB	POINTER			
			0A	02	FB	001C1	CALLS	#2, FETCH_INSTRUCTION			
		7E	69	56	E9	001C6	BLBC	R6, 20\$		0899	
		0000V	CF	08	AE	001C9	ADDL3	POINTER, OPERAND VALUE, -(SP)		0900	
				02	01	FB	CALLS	#1, PRINT_ADDRESS			
				F3	31	001D3	BRW	71\$		0897	
				7E	D4	001D6	CLRL	-(SP)		0908	
				56	DD	001D8	PUSHL	R6			
7E	09	A4	04	04	EF	001DA	EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)			
			6A	14	9F	001E0	PUSHAB	POINTER			
				04	FB	001E3	CALLS	#4, FETCH_OPERAND			
				52	D7	001E6	DECL	STATE		0909	
				56	11	001E8	BRB	30\$		0886	
				7E	D4	001EA	CLRL	-(SP)		0919	
				56	DD	001EC	PUSHL	R6			
7E	09	A4	04	04	EF	001EE	EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)			
			6A	14	9F	001F4	PUSHAB	POINTER			
				04	FB	001F7	CALLS	#4, FETCH_OPERAND			
				41	11	001FA	BRB	29\$		0920	
				7E	D4	001FC	CLRL	-(SP)		0925	
				56	DD	001FE	PUSHL	R6			
7E	09	A4	04	04	EF	00200	EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)			
				5F	11	00206	BRB	33\$			
				7E	D4	00208	CLRL	-(SP)		0932	
				56	DD	0020A	PUSHL	R6			
7E	09	A4	04	04	EF	0020C	EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)			
			6A	14	9F	00212	PUSHAB	POINTER			
			53	04	FB	00215	CALLS	#4, FETCH_OPERAND		0933	
				52	D0	00218	MOVL	STATE, COUNT			
			0A	1B	11	0021B	BRB	28\$		0935	
				56	E9	0021D	BLBC	R6, 27\$			
				04	9F	00220	PUSHAB	DELIMITER			
				01	DD	00223	PUSHL	#1			
				57	DD	00225	PUSHL	R7			
			68	03	FB	00227	CALLS	#3, DBGSPRINT			
				7E	D4	0022A	CLRL	-(SP)		0936	
				56	DD	0022C	PUSHL	R6			
				7E	D4	0022E	CLRL	-(SP)			
			6A	14	9F	00230	PUSHAB	POINTER			
				04	FB	00233	CALLS	#4, FETCH_OPERAND			
			0C	53	D7	00236	DECL	COUNT		0933	
				53	D1	00238	CMPL	COUNT, #12			
			52	E0	18	0023B	BGEQ	26\$			
				01	D0	0023D	MOVL	#1, STATE		0938	
				60	11	00240	BRB	36\$		0886	
7E	09	A4	04	00	EF	00242	EXTZV	#0, #4, 9(OPCODE_ENTRY), -(SP)		0943	
				0C	9F	00248	PUSHAB	POINTER			
		0000V	CF	02	FB	0024B	CALLS	#2, FETCH_INSTRUCTION			
			80	56	E9	00250	BLBC	R6, 20\$		0944	
7E	09	A4	04	00	EF	00253	EXTZV	#0, #4, 9(OPCODE_ENTRY), -(SP)			
		0000V	CF	01	FB	00259	CALLS	#1, PRINT_OPERAND			
				02	31	0025E	BRW	71\$		0942	
				7E	D4	00261	CLRL	-(SP)		0949	

				56	DD	00263	PUSHL	R6			
				7E	D4	00265	CLRL	-(SP)			
			14	AE	9F	00267	PUSHAB	POINTER			
	6A			04	FB	0026A	CALLS	#4, FETCH_OPERAND			
	52			02	DD	0026D	MOVL	#2, STATE			0950
				30	11	00270	BRB	36\$			0886
				7E	D4	00272	CLRL	-(SP)			0955
				56	DD	00274	PUSHL	R6			
7E		09	A4	04	00	EF	EXTZV	#0, #4, 9(OPCODE_L_TRY), -(SP)			
			14	AE	9F	0027C	PUSHAB	POINTER			
	6A			04	FB	0027F	CALLS	#4, FETCH_OPERAND			
	0A			56	E9	00282	BLBC	R6, 35\$			0956
			04	AE	9F	00285	PUSHAB	DELIMITER			
				01	DD	00288	PUSHL	#1			
				57	DD	0028A	PUSHL	R7			
	68			03	FB	0028C	CALLS	#3, DBG\$PRINT			
				7E	D4	0028F	CLRL	-(SP)			0957
				56	DD	00291	PUSHL	R6			
7E		09	A4	04	EF	00293	EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)			
			14	AE	9F	00299	PUSHAB	POINTER			
	6A			04	FB	0029C	CALLS	#4, FETCH_OPERAND			
	52			03	DD	0029F	MOVL	#3, STATE			0958
				FEB0	31	002A2	BRW	14\$			0886
				7E	D4	002A5	CLRL	-(SP)			0963
				56	DD	002A7	PUSHL	R6			
				7E	D4	002A9	CLRL	-(SP)			
			14	AE	9F	002AB	PUSHAB	POINTER			
	6A			04	FB	002AE	CALLS	#4, FETCH_OPERAND			
	0A			56	E9	002B1	BLBC	R6, 38\$			0964
			04	AE	9F	002B4	PUSHAB	DELIMITER			
				01	DD	002B7	PUSHL	#1			
				57	DD	002B9	PUSHL	R7			
	68			03	FB	002BB	CALLS	#3, DBG\$PRINT			
				7E	D4	002BE	CLRL	-(SP)			0965
				56	DD	002C0	PUSHL	R6			
				02	DD	002C2	PUSHL	#2			
			14	AE	9F	002C4	PUSHAB	POINTER			
	6A			04	FB	002C7	CALLS	#4, FETCH_OPERAND			
	0A			56	E9	002CA	BLBC	R6, 39\$			0966
			04	AE	9F	002CD	PUSHAB	DELIMITER			
				01	DD	002D0	PUSHL	#1			
				57	DD	002D2	PUSHL	R7			
	68			03	FB	002D4	CALLS	#3, DBG\$PRINT			
				7E	D4	002D7	CLRL	-(SP)			0967
				56	DD	002D9	PUSHL	R6			
				0B	DD	002DB	PUSHL	#11			
				51	11	002DD	BRB	44\$			
				7E	D4	002DF	CLRL	-(SP)			0973
				56	DD	002E1	PUSHL	R6			
				02	DD	002E3	PUSHL	#2			
			14	AE	9F	002E5	PUSHAB	POINTER			
	6A			04	FB	002E8	CALLS	#4, FETCH_OPERAND			
	0A			56	E9	002EB	BLBC	R6, 41\$			0974
			04	AE	9F	002EE	PUSHAB	DELIMITER			
				01	DD	002F1	PUSHL	#1			
				57	DD	002F3	PUSHL	R7			
	68			03	FB	002F5	CALLS	#3, DBG\$PRINT			

		7E	D4	002F8	41\$:	CLRL	-(SP)	0975
		56	DD	002FA		PUSHL	R6	
		02	DD	002FC		PUSHL	#2	
6A	14	AE	9F	002FE		PUSHAB	POINTER	
0A		04	FB	00301		CALLS	#4, FETCH_OPERAND	
		56	E9	00304		BLBC	R6, 42\$	0976
	04	AE	9F	00307		PUSHAB	DELIMITER	
		01	DD	0030A		PUSHL	#1	
		57	DD	0030C		PUSHL	R7	
68		03	FB	0030E		CALLS	#3, DBG\$PRINT	
		7E	D4	00311	42\$:	CLRL	-(SP)	0977
		56	DD	00313		PUSHL	R6	
		02	DD	00315		PUSHL	#2	
6A	14	AE	9F	00317		PUSHAB	POINTER	
0A		04	FB	0031A		CALLS	#4, FETCH_OPERAND	
		56	E9	0031D		BLBC	R6, 43\$	0978
	04	AE	9F	00320		PUSHAB	DELIMITER	
		01	DD	00323		PUSHL	#1	
		57	DD	00325		PUSHL	R7	
68		03	FB	00327		CALLS	#3, DBG\$PRINT	
		7E	D4	0032A	43\$:	CLRL	-(SP)	0979
		56	DD	0032C		PUSHL	R6	
		03	DD	0032E		PUSHL	#3	
		51	11	00330	44\$:	BRB	49\$	
		7E	D4	00332	45\$:	CLRL	-(SP)	0985
		56	DD	00334		PUSHL	R6	
		02	DD	00336		PUSHL	#2	
6A	14	AE	9F	00338		PUSHAB	POINTER	
0A		04	FB	0033B		CALLS	#4, FETCH_OPERAND	
		56	E9	0033E		BLBC	R6, 46\$	0986
	04	AE	9F	00341		PUSHAB	DELIMITER	
		01	DD	00344		PUSHL	#1	
		57	DD	00346		PUSHL	R7	
68		03	FB	00348		CALLS	#3, DBG\$PRINT	
		7E	D4	0034B	46\$:	CLRL	-(SP)	0987
		56	DD	0034D		PUSHL	R6	
		03	DD	0034F		PUSHL	#3	
6A	14	AE	9F	00351		PUSHAB	POINTER	
0A		04	FB	00354		CALLS	#4, FETCH_OPERAND	
		56	E9	00357		BLBC	R6, 47\$	0988
	04	AE	9F	0035A		PUSHAB	DELIMITER	
		01	DD	0035D		PUSHL	#1	
		57	DD	0035F		PUSHL	R7	
68		03	FB	00361		CALLS	#3, DBG\$PRINT	
		7E	D4	00364	47\$:	CLRL	-(SP)	0989
		56	DD	00366		PUSHL	R6	
		02	DD	00368		PUSHL	#2	
6A	14	AE	9F	0036A		PUSHAB	POINTER	
0A		04	FB	0036D		CALLS	#4, FETCH_OPERAND	
		56	E9	00370		BLBC	R6, 48\$	0990
	04	AE	9F	00373		PUSHAB	DELIMITER	
		01	DD	00376		PUSHL	#1	
		57	DD	00378		PUSHL	R7	
68		03	FB	0037A		CALLS	#3, DBG\$PRINT	
		7E	D4	0037D	48\$:	CLRL	-(SP)	0991
		56	DD	0037F		PUSHL	R6	
		02	DD	00381		PUSHL	#2	



53		0130	31	00383	49\$:	BRW	70\$		
		05	00	00386	50\$:	MOVL	#5, COUNT		0996
		7E	04	00389	51\$:	CLRL	-(SP)		0998
		56	DD	0038B		PUSHL	R6		
		02	DD	0038D		PUSHL	#2		
	14	AE	9F	0038F		PUSHAB	POINTER		
6A		04	FB	00392		CALLS	#4, FETCH_OPERAND		
OE		56	E9	00395		BLBC	R6, 52\$		0999
		53	D5	00398		TSTL	COUNT		
		0A	13	0039A		BEQL	52\$		
	04	AE	9F	0039C		PUSHAB	DELIMITER		
		01	DD	0039F		PUSHL	#1		
		57	DD	003A1		PUSHL	R7		
68		03	FB	003A3		CALLS	#3, DBG\$PRINT		
E0		53	F4	003A6	52\$:	SOBGEQ	COUNT, 51\$		0996
		71	11	003A9		BRB	60\$		0995
		7E	04	003AB	53\$:	CLRL	-(SP)		1008
		56	DD	003AD		PUSHL	R6		
53	09	A4	00	EF	003AF	EXTZV	#0, #4, 9(OPCODE_ENTRY), R3		
		53	DD	003B5		PUSHL	R3		
	14	AE	9F	003B7		PUSHAB	POINTER		
6A		04	FB	003BA		CALLS	#4, FETCH_OPERAND		
OA		56	E9	003BD		BLBC	R6, 54\$		1009
	04	AE	9F	003C0		PUSHAB	DELIMITER		
		01	DD	003C3		PUSHL	#1		
		57	DD	003C5		PUSHL	R7		
68		03	FB	003C7		CALLS	#3, DBG\$PRINT		
		7E	04	003CA	54\$:	CLRL	-(SP)		1010
	0048	8F	BB	003CC		PUSHR	#*M<R3,R6>		
	14	AE	9F	003D0		PUSHAB	POINTER		
6A		04	FB	003D3		CALLS	#4, FETCH_OPERAND		
OA		56	E9	003D6		BLBC	R6, 55\$		1011
	04	AE	9F	003D9		PUSHAB	DELIMITER		
		01	DD	003DC		PUSHL	#1		
		57	DD	003DE		PUSHL	R7		
68		03	FB	003E0		CALLS	#3, DBG\$PRINT		
		7E	04	003E3	55\$:	CLRL	-(SP)		1016
	0048	8F	BB	003E5		PUSHR	#*M<R3,R6>		
	14	AE	9F	003E9		PUSHAB	POINTER		
6A		04	FB	003EC		CALLS	#4, FETCH_OPERAND		
		53	D5	003EF		TSTL	R3		1023
		05	12	003F1		BNEQ	56\$		
50		69	9A	003F3		MOVZBL	OP_BUFFER, LIMIT		
		17	11	003F6		BRB	59\$		
01		53	D1	003F8	56\$:	CMPL	R3, #1		1024
		05	12	003FB		BNEQ	57\$		
50		69	3C	003FD		MOVZWL	OP_BUFFER, LIMIT		
		0D	11	00400		BRB	59\$		
02		53	D1	00402	57\$:	CMPL	R3, #2		1025
		05	13	00405		BEQL	58\$		
50		01	CE	00407		MNEGL	#1, LIMIT		
		03	11	0040A		BRB	59\$		
50		69	D0	0040C	58\$:	MOVL	OP_BUFFER, LIMIT		
OC		56	E8	0040F	59\$:	BLBS	R6, 61\$		1030
08		AE	3E	00412		MOVAW	@POINTER[LIMIT], POINTER		
08		02	C0	00418		ADDL2	#2, POINTER		
		42	11	0041C	60\$:	BRB	65\$		

		55	08	AE	D0	0041E	618:	MOVL	POINTER, START	1034
		53	01	A0	9E	00422		MOVAB	1(R0), COUNT	1043
				35	11	00426		BRB	648	
	09	00000000G	00	01	E1	00428	628:	BBC	#1, DBG\$GV_CONTROL+1, 638	1038
				8F	DD	00430		PUSHL	#164072	
		6B		01	FB	00436		CALLS	#1, LIB\$SIGNAL	
				01	DD	00439	638:	PUSHL	#1	1040
			0C	AE	9F	0043B		PUSHAB	POINTER	
	0000V	CF		02	FB	0043E		CALLS	#2, FETCH_INSTRUCTION	
	00000000G	00		00	FB	00443		CALLS	#0, DBG\$NEWLINE	1041
			18	A7	9F	0044A		PUSHAB	P, AAH	1042
				02	DD	0044D		PUSHL	#2	
				57	DD	0044F		PUSHL	R7	
		6B		03	FB	00451		CALLS	#3, DBG\$PRINT	
			00	B9	9F	00454		PUSHAB	CASE OFFSET[START]	1043
	0000V	CF		01	FB	00458		CALLS	#1, PRINT_ADDRESS	
		C8		53	F4	0045D	648:	SOBGEQ	COUNT, 628	1035
				67	11	00460	658:	BRB	718	1005
				7E	D4	00462	668:	CLRL	-(SP)	1051
				56	DD	00464		PUSHL	R6	
7E	09	A4	04	04	EF	00466		EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)	
			14	AE	9F	0046C		PUSHAB	POINTER	
		6A		04	FB	0046F		CALLS	#4, FETCH_OPERAND	
		0A		56	E9	00472		BLBC	R6, 678	1052
			04	AE	9F	00475		PUSHAB	DELIMITER	
				01	DD	00478		PUSHL	#1	
				57	DD	0047A		PUSHL	R7	
		6B		03	FB	0047C		CALLS	#3, DBG\$PRINT	
				7E	D4	0047F	678:	CLRL	-(SP)	1053
				56	DD	00481		PUSHL	R6	
7E	09	A4	04	00	EF	00483		EXTZV	#0, #4, 9(OPCODE_ENTRY), -(SP)	
			14	AE	9F	00489		PUSHAB	POINTER	
		6A		04	FB	0048C		CALLS	#4, FETCH_OPERAND	
		0A		56	E9	0048F		BLBC	R6, 688	1054
			04	AE	9F	00492		PUSHAB	DELIMITER	
				01	DD	00495		PUSHL	#1	
				57	DD	00497		PUSHL	R7	
		6B		03	FB	00499		CALLS	#3, DBG\$PRINT	
				7E	D4	0049C	688:	CLRL	-(SP)	1055
				56	DD	0049E		PUSHL	R6	
7E	09	A4	04	04	EF	004A0		EXTZV	#4, #4, 9(OPCODE_ENTRY), -(SP)	
			14	AE	9F	004A6		PUSHAB	POINTER	
		6A		04	FB	004A9		CALLS	#4, FETCH_OPERAND	
		0A		56	E9	004AC		BLBC	R6, 698	1056
			04	AE	9F	004AF		PUSHAB	DELIMITER	
				01	DD	004B2		PUSHL	#1	
				57	DD	004B4		PUSHL	R7	
		6B		03	FB	004B6		CALLS	#3, DBG\$PRINT	
				7E	D4	004B9	698:	CLRL	-(SP)	1057
				56	DD	004BB		PUSHL	R6	
7E	09	A4	04	00	EF	004BD		EXTZV	#0, #4, 9(OPCODE_ENTRY), -(SP)	
			14	AE	9F	004C3	708:	PUSHAB	POINTER	
		6A		04	FB	004C6		CALLS	#4, FETCH_OPERAND	
		50		08	AE	D0	718:	MOVL	POINTER, R0	1067
					04	004CD		RET		1068
					0000	004CE	728:	.WORD	Save nothing	0786
		50		08	AC	D0		MOVL	8(AP), R0	

DBGENCDEC  
V04-000

M 6  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 57  
(17)

	50	04	A0	D0	004D4	MOVL	4(R0), R0	
		F8	A0	9F	004D8	PUSHAB	ERROR_VALUE	
		FC	A0	9F	004DB	PUSHAB	SIGNAL_FLAG	
			02	DD	004DE	PUSHL	#2	
			5E	DD	004E0	PUSHL	SP	
	7E		AC	7D	004E2	MOVQ	4(AP), -(SP)	
FAE1	CF		03	FB	004E6	CALLS	#3, DECODE_HANDLER	
			04	04	004EB	RET		

; Routine Size: 1260 bytes,    Routine Base: DBG\$CODE + 0034

```

953 1069 1 GLOBAL ROUTINE DBG$Ins_Encode(Input_Buffer,Output_Buffer,Relocation) =
954 1070 BEGIN
955 1071 MAP Input_Buffer : REF VECTOR [,BYTE]; ! %ASCII String
956 1072 Output_Buffer : REF VECTOR [,BYTE]; ! Encoded instruction
957 1073 LOCAL
958 1074 Operand_number : INITIAL( 0 ); ! A007
959 1075 Encode : BLOCK [20,BYTE] FIELD(Encode_Fields),
960 1076 Local_Buffer : VECTOR[256,BYTE]
961 1077 Opcode_Entry : REF BLOCK[10,BYTE] FIELD(Opcode_Entry_Fields),
962 1078 State;
963 1079
964 1080 Encode[Enc_Input_Class] = dsc$tk_class_s;
965 1081 Encode[Enc_Input_Dtype] = dsc$tk_dtype_t;
966 1082 Encode[Enc_Input_Length] = .Input_Buffer[0];
967 1083 Encode[Enc_Input_Buffer] = Local_Buffer[0];
968 1084 Encode[Enc_Output_Length] = 0;
969 1085 Encode[Enc_Output_Buffer] = Output_Buffer[1];
970 1086 Encode[Enc_Final_Address] = .Relocation;
971 1087
972 1088 ch$move(.Input_Buffer[0],Input_Buffer[1],Local_Buffer[0]);
973 1089 Opcode_Entry = Opcode_Name_Index(Encode[Enc_Input_Desc],%C' '); ! Changed to call Opcode_Name
974 1090 ! instead of DBG$OPCODE_INDEX
975 1091 Opcode_Entry = DBG$Opcode_Name_Table[.Opcode_Entry,offset];
976 1092
977 1093 IF .Opcode_Entry[op_code_one] NEQ 0 THEN
978 1094 Store_Operand(Encode,Opcode_Entry[op_code_one],1);
979 1095 Store_Operand(Encode,Opcode_Entry[op_code_two],1);
980 1096
981 1097 State = .Opcode_Entry[op_kind];
982 1098
983 1099 WHILE (.State NEQ simple_0_operand) DO
984 1100 BEGIN
985 1101 CASE .State FROM simple_1_operand TO maximum_state OF
986 1102 SET
987 1103
988 1104 [simple_1_operand,
989 1105 simple_2_operand,
990 1106 simple_3_operand]:
991 1107 BEGIN
992 1108 Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);! M007
993 1109 State = .State - 1;
994 1110 END;
995 1111
996 1112 [branch_0_operand]:
997 1113 BEGIN
998 1114 LOCAL Address,Length;
999 1115 Operand_number = .Operand_number + 1; ! A007
1000 1116 Length = Scan_Operand(Encode[Enc_Input_Desc],%C' ');
1001 1117 IF NOT Parse_Expression(-1,.Encode[Enc_Input_Buffer],
1002 1118 .Encode[Enc_Input_Length],Address) THEN SIGNAL(dbg$_INVEXPR,1,.Operand_number); ! MO
1003 1119 IF (Check_Register(.Address) GEQ 0) THEN SIGNAL(dbg$_INVEXPR,1,.Operand_number); ! MO
1004 1120 Encode[Enc_Input_Buffer] = .Encode[Enc_Input_Buffer] + .Length;
1005 1121 Encode[Enc_Input_Length] = .Encode[Enc_Input_Length] - .Length;
1006 1122
1007 1123 Length = .Data_Size[.Opcode_Entry[op_type_one]];
1008 1124
1009 1125 Address = .Address - (.Length + .Encode[Enc_Final_Address])

```



```
.. 1010 1126 4
.. 1011 1127 4
.. 1012 1128 4
.. 1013 1129 4
.. 1014 1130 4
.. 1015 1131 4
.. 1016 1132 4
.. 1017 1133 4
.. 1018 1134 4
.. 1019 1135 4
.. 1020 1136 4
.. 1021 1137 4
.. 1022 1138 4
.. 1023 1139 4
.. 1024 1140 4
.. 1025 1141 4
.. 1026 1142 4
.. 1027 1143 4
.. 1028 1144 4
.. 1029 1145 4
.. 1030 1146 4
.. 1031 1147 4
.. 1032 1148 4
.. 1033 1149 4
.. 1034 1150 4
.. 1035 1151 4
.. 1036 1152 4
.. 1037 1153 4
.. 1038 1154 4
.. 1039 1155 4
.. 1040 1156 4
.. 1041 1157 4
.. 1042 1158 4
.. 1043 1159 4
.. 1044 1160 4
.. 1045 1161 4
.. 1046 1162 4
.. 1047 1163 4
.. 1048 1164 4
.. 1049 1165 4
.. 1050 1166 4
.. 1051 1167 4
.. 1052 1168 4
.. 1053 1169 4
.. 1054 1170 4
.. 1055 1171 4
.. 1056 1172 4
.. 1057 1173 4
.. 1058 1174 4
.. 1059 1175 4
.. 1060 1176 4
.. 1061 1177 4
.. 1062 1178 4
.. 1063 1179 4
.. 1064 1180 4
.. 1065 1181 4
.. 1066 1182 4

Store_Operand(Encode,Address,.Length);
EXITLOOP;
END;

[branch_1_operand,
branch_2_operand,
branch_3_operand]:
BEGIN
Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);! M007
State = .State - 1;
END;

[convert_datatype,
evaluate_address,
simple_bit_field,
routine_dispatch,
polynomial_value,
probe_for_access,
string_3_operand]:
BEGIN
Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);! M007
State = Simple_1_Operand;
END;

[string_4_operand]:
BEGIN
Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);! M007
State = Simple_2_Operand;
END;

[string_5_operand,
string_6_operand]:
BEGIN
Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);! M007
DECR Count FROM .State TO string_5_operand DO
Parse_Operand(Encode,context_b,Operand_number);! M007
State = Simple_1_Operand;
END;

[trailing_operand]:
BEGIN
Store_Operand(Encode,.Opcode_Entry[op_type_one]);
IF .Print_Flag THEN Print_Operand(.Opcode_Entry[op_type_one]);
EXITLOOP;
END;

[complex_SHIFT]:
BEGIN
Parse_Operand(Encode,context_b,Operand_number);! M007
State = Simple_2_Operand;
END;

[complex_EMOD]:
BEGIN
Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);! M007
Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);! M007
```

```
1067      1183      4      State = Simple_3_Operand;
1068      1184      4      END;
1069      1185      4
1070      1186      4      [complex_CRC]:
1071      1187      4      BEGIN
1072      1188      4      Parse_Operand(Encode,context_b,Operand_number);      ! M007
1073      1189      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1074      1190      4      Parse_Operand(Encode,context_t,Operand_number);      ! M007
1075      1191      4      EXITLOOP;
1076      1192      4      END;
1077      1193      4
1078      1194      4      [complex_EMUL]:
1079      1195      4      BEGIN
1080      1196      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1081      1197      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1082      1198      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1083      1199      4      Parse_Operand(Encode,context_q,Operand_number);      ! M007
1084      1200      4      EXITLOOP;
1085      1201      4      END;
1086      1202      4
1087      1203      4      [complex_EDIV]:
1088      1204      4      BEGIN
1089      1205      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1090      1206      4      Parse_Operand(Encode,context_q,Operand_number);      ! M007
1091      1207      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1092      1208      4      Parse_Operand(Encode,context_l,Operand_number);      ! M007
1093      1209      4      EXITLOOP;
1094      1210      4      END;
1095      1211      4
1096      1212      4      [complex_INDEX]:
1097      1213      4      BEGIN
1098      1214      4      DECR count FROM 5 TO 0 DO Parse_Operand(Encode,context_l,Operand_number);! M007
1099      1215      4      EXITLOOP;
1100      1216      4      END;
1101      1217      4
1102      1218      4      [complex_CASE]:
1103      1219      4      BEGIN
1104      1220      4      Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);! M007
1105      1221      4      Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);! M007
1106      1222      4      Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);! M007
1107      1223      4      EXITLOOP;
1108      1224      4      END;
1109      1225      4
1110      1226      4      [complex_ASHP]:
1111      1227      4      BEGIN
1112      1228      4      Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);!
1113      1229      4      Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);!
1114      1230      4      Parse_Operand(Encode,.Opcode_Entry[op_type_two],Operand_number);!
1115      1231      4      Parse_Operand(Encode,.Opcode_Entry[op_type_one],Operand_number);!
1116      1232      4      EXITLOOP;
1117      1233      4      END;
1118      1234      4
1119      1235      4      [INRANGE,OUTRANGE]:
1120      1236      4      $DBG_ERROR('DBG$Encode_Decode - bad opcode table entry');
1121      1237      4
1122      1238      4      TES;
1123      1239      4      END;
```

```
.PSECT DBGSPLIT,NOWRT, SHR, PIC.0
```

```
.PSECT DBG$CODE,NOWRT, SHR, PIC.0
```

```
.ENTRY DBG$INS_ENCODE, Save R2,R3,R4,R5,R6,R7,R8,- : 1069
```

			59	0000V	CF	9E	00002	MOVAB	STORE_OPERAND, R9		
			58	00000000G	00	9E	00007	MOVAB	LIB\$SIGNAL, R8		
			57	000000000'	EF	9E	0000E	MOVAB	DBG\$OPCODE_NAME_TABLE, R7		
			56	0000V	CF	9E	00015	MOVAB	PARSE_OPERAND, R6		
			5E	FEE4	CE	9E	0001A	MOVAB	-284(SP), SP		
				04	AE	D4	0001F	CLRL	OPERAND_NUMBER		1070
	EE	AD		01OE	8F	B0	00022	MOVW	#270, ENCODE+2		1081
		50		04	AC	D0	00028	MOVL	INPUT_BUFFER, R0		1082
	EC	AD			60	9B	0002C	MOVZBW	(R0), -ENCODE		
	FO	AD		08	AE	9E	00030	MOVAB	LOCAL_BUFFER, ENCODE+4		1083
				F4	AD	D4	00035	CLRL	ENCODE+8		1084
F8	AD	08	AC		01	C1	00038	ADDL3	#1, OUTPUT_BUFFER, ENCODE+12		1085
		FC	AD	OC	AC	D0	0003E	MOVL	RELOCATION, ENCODE+16		1086
OB	AE	01	S1		60	9A	00043	MOVZBL	(R0) R1		1088
			A0		S1	28	00046	MOVCS	R1, 1(R0), LOCAL_BUFFER		
					20	DD	0004C	PUSHL	#32		1089
				EC	AD	9F	0004E	PUSHAB	ENCODE		
		0000V	CF		02	FB	00051	CALLS	#2, OPCODE_NAME_INDEX		
			54		50	D0	00056	MOVL	R0, OPCODE_ENTRY		
	50		54		0A	C5	00059	MULL3	#10, OPCODE_ENTRY, R0		1091
	54		50		57	C1	0005D	ADDL3	R7, R0, OPCODE_ENTRY		
				06	A4	95	00061	TSTB	6(OPCODE_ENTRY)		1093
					0B	13	00064	BEQL	1\$		
					01	DD	00066	PUSHL	#1		1094
				06	A4	9F	00068	PUSHAB	6(OPCODE_ENTRY)		
				EC	AD	9F	0006B	PUSHAB	ENCODE		
		69			03	FB	0006E	CALLS	#3, STORE_OPERAND		1095
					01	DD	00071	PUSHL	#1		
				07	A4	9F	00073	PUSHAB	7(OPCODE_ENTRY)		
				EC	AD	9F	00076	PUSHAB	ENCODE		
		69			03	FB	00079	CALLS	#3, STORE_OPERAND		
		53		08	A4	9A	0007C	MOVZBL	8(OPCODE_ENTRY), STATE		1097
					53	D5	00080	TSTL	STATE		1099
					03	12	00082	BNEQ	3\$		
					022C	31	00084	BRW	32\$		
					53	CF	00087	CASEL	STATE, #1, #22		1101
16			01		003F		00088	.WORD	6\$-4\$, -		
003F			003F		003F						

00D7  
00F3  
011E  
016F

00C4  
00E8  
002E  
012E  
01EC

00C4  
00D7  
002E  
0197  
0152

00C4  
0228  
00F3  
01D2  
01BF

00093  
0009B  
000A3  
000AB  
000B3

61-48,-  
65-48,-  
73-48,-  
101-48,-  
101-48,-  
101-48,-  
128-48,-  
328-48,-  
128-48,-  
158-48,-  
148-48,-  
148-48,-  
58-48,-  
58-48,-  
188-48,-  
288-48,-  
248-48,-  
208-48,-  
238-48,-  
268-48,-  
228-48,-  
298-48,-

			114D	C7	9F	000B9	58:	PUSHAB	P.AAJ	1236
				01	DD	000BD		PUSHL	#1	
			00028362	8F	DD	000BF		PUSHL	#164706	
		68		03	FB	000C5		CALLS	#3, LIBSSIGNAL	
				B6	11	000C8		BRB	28	
			04	AE	9F	000CA	68:	PUSHAB	OPERAND_NUMBER	1108
7E	09	A4	04	00	EF	000CD		EXTZV	#0, #4, '9(OPCODE_ENTRY), -(SP)	
				0082	31	000D3		BRW	118	
			04	AE	D6	000D6	78:	INCL	OPERAND_NUMBER	1115
				2C	DD	000D9		PUSHL	#44	1116
			EC	AD	9F	000DB		PUSHAB	ENCODE	
0000V	CF			02	FB	000DE		CALLS	#2, SCAN_OPERAND	
	52			50	DD	000E3		MOVL	RO, LENGTH	
				5E	DD	000E6		PUSHL	SP	1117
	7E		EC	AD	3C	000E8		MOVZWL	ENCODE, -(SP)	1118
			FO	AD	DD	000EC		PUSHL	ENCODE+4	1117
	7E			01	CE	000EF		MNEGL	#1, -(SP)	
0000V	CF			04	FB	000F2		CALLS	#4, PARSE_EXPRESSION	
	0E			50	EB	000F7		BLBS	RO, 88	
			04	AE	DD	000FA		PUSHL	OPERAND_NUMBER	1118
				01	DD	000FD		PUSHL	#1	
			00028290	8F	DD	000FF		PUSHL	#164496	
	68			03	FB	00105		CALLS	#3, LIBSSIGNAL	
0000V	CF			6E	DD	00108	88:	PUSHL	ADDRESS	1119
				01	FB	0010A		CALLS	#1, CHECK_REGISTER	
				50	D5	0010F		TSTL	RO	
				0E	19	00111		BLSS	98	
			04	AE	DD	00113		PUSHL	OPERAND_NUMBER	
				01	DD	00116		PUSHL	#1	
			00028290	8F	DD	00118		PUSHL	#164496	
	68			03	FB	0011E		CALLS	#3, LIBSSIGNAL	
FO	AD			52	C0	00121	92:	ADDL2	LENGTH, ENCODE+4	1120
EC	AD			52	A2	00125		BRW2	LENGTH, ENCODE	1121
	04			00	EF	00129		BRW2	#0, #4, '9(OPCODE_ENTRY), RO	1123
50	09	A4	52	10A0	C740	9A	0012F	BRW2	DATA_SIZE[RO], LENGTH	



		50	52	FC	AD	C1	00135	ADDL3	ENCODE+16, LENGTH, R0	1125
		50	50	F4	AD	C0	0013A	ADDL2	ENCODE+8, R0	1126
		6E	6E		50	C2	0013E	SUBL2	R0, ADDRESS	1125
					52	DD	00141	PUSHL	LENGTH	1127
				04	AE	9F	00143	PUSHAB	ADDRESS	
				EC	AD	9F	00146	PUSHAB	ENCODE	
		69			03	FB	00149	CALLS	#3, STORE_OPERAND	
					0164	31	0014C	BRW	32\$	1113
7E	09	A4	04	04	AE	9F	0014F	PUSHAB	OPERAND_NUMBER	1135
			04	04	EF	00152	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)		
			66	EC	AD	9F	00158	PUSHAB	ENCODE	
					03	FB	0015B	CALLS	#3, PARSE_OPERAND	
					53	D7	0015E	DECL	STATE	1136
					78	11	00160	BRB	21\$	1101
7E	09	A4	04	04	AE	9F	00162	PUSHAB	OPERAND_NUMBER	1147
			04	04	EF	00165	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)		
			66	EC	AD	9F	0016B	PUSHAB	ENCODE	
					03	FB	0016E	CALLS	#3, PARSE_OPERAND	
					31	11	00171	BRB	17\$	1148
7E	09	A4	04	04	AE	9F	00173	PUSHAB	OPERAND_NUMBER	1153
			04	04	EF	00176	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)		
					30	11	0017C	BRB	19\$	
7E	09	A4	04	04	AE	9F	0017E	PUSHAB	OPERAND_NUMBER	1160
			04	04	EF	00181	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)		
			66	EC	AD	9F	00187	PUSHAB	ENCODE	
			52		03	FB	0018A	CALLS	#3, PARSE_OPERAND	
					53	D0	0018D	MOVL	STATE, COUNT	1161
					0D	11	00190	BRB	16\$	
				04	AE	9F	00192	PUSHAB	OPERAND_NUMBER	1162
					7E	D4	00195	CLRL	-(SP)	
			66	EC	AD	9F	00197	PUSHAB	ENCODE	
					03	FB	0019A	CALLS	#3, PARSE_OPERAND	
			0C		52	D7	0019D	DECL	COUNT	
			53		52	D1	0019F	CMPL	COUNT, #12	
					EE	18	001A2	BGEQ	15\$	
					01	D0	001A4	MOVL	#1, STATE	1163
					31	11	001A7	BRB	21\$	1101
				04	AE	9F	001A9	PUSHAB	OPERAND_NUMBER	1175
					7E	D4	001AC	CLRL	-(SP)	
			66	EC	AD	9F	001AE	PUSHAB	ENCODE	
			53		03	FB	001B1	CALLS	#3, PARSE_OPERAND	
					02	D0	001B4	MOVL	#2, STATE	1176
					21	11	001B7	BRB	21\$	1101
7E	09	A4	04	04	AE	9F	001B9	PUSHAB	OPERAND_NUMBER	1181
			04	00	EF	001BC	EXTZV	#0, #4, -9(_OPCODE_ENTRY), -(SP)		
			66	EC	AD	9F	001C2	PUSHAB	ENCODE	
					03	FB	001C5	CALLS	#3, PARSE_OPERAND	
7E	09	A4	04	04	AE	9F	001C8	PUSHAB	OPERAND_NUMBER	1182
			04	04	EF	001CB	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)		
			66	EC	AD	9F	001D1	PUSHAB	ENCODE	
			53		03	FB	001D4	CALLS	#3, PARSE_OPERAND	
					03	D0	001D7	MOVL	#3, STATE	1183
					FEA3	31	001DA	BRW	2\$	1101
				04	AE	9F	001DD	PUSHAB	OPERAND_NUMBER	1188
					7E	D4	001E0	CLRL	-(SP)	
			66	EC	AD	9F	001E2	PUSHAB	ENCODE	
					03	FB	001E5	CALLS	#3, PARSE_OPERAND	

			04	AE	9F	001E8	PUSHAB	OPERAND_NUMBER	1189
				02	DD	001EB	PUSHL	#2	
		66	EC	AD	9F	001ED	PUSHAB	ENCODE	
				03	FB	001F0	CALLS	#3, PARSE OPERAND	
			04	AE	9F	001F3	PUSHAB	OPERAND_NUMBER	1190
				0B	DD	001F6	PUSHL	#11	
				4E	11	001F8	BRB	25\$	
			04	AE	9F	001FA	PUSHAB	OPERAND_NUMBER	1196
				02	DD	001FD	PUSHL	#2	
		66	EC	AD	9F	001FF	PUSHAB	ENCODE	
				03	FB	00202	CALLS	#3, PARSE OPERAND	
			04	AE	9F	00205	PUSHAB	OPERAND_NUMBER	1197
				02	DD	00208	PUSHL	#2	
		66	EC	AD	9F	0020A	PUSHAB	ENCODE	
				03	FB	0020D	CALLS	#3, PARSE OPERAND	
			04	AE	9F	00210	PUSHAB	OPERAND_NUMBER	1198
				02	DD	00213	PUSHL	#2	
		66	EC	AD	9F	00215	PUSHAB	ENCODE	
				03	FB	00218	CALLS	#3, PARSE OPERAND	
			04	AE	9F	0021B	PUSHAB	OPERAND_NUMBER	1199
				03	DD	0021E	PUSHL	#3	
				26	11	00220	BRB	25\$	
			04	AE	9F	00222	PUSHAB	OPERAND_NUMBER	1205
				02	DD	00225	PUSHL	#2	
		66	EC	AD	9F	00227	PUSHAB	ENCODE	
				03	FB	0022A	CALLS	#3, PARSE OPERAND	
			04	AE	9F	0022D	PUSHAB	OPERAND_NUMBER	1206
				03	DD	00230	PUSHL	#3	
		66	EC	AD	9F	00232	PUSHAB	ENCODE	
				03	FB	00235	CALLS	#3, PARSE OPERAND	
			04	AE	9F	00238	PUSHAB	OPERAND_NUMBER	1207
				02	DD	0023B	PUSHL	#2	
		66	EC	AD	9F	0023D	PUSHAB	ENCODE	
				03	FB	00240	CALLS	#3, PARSE OPERAND	
			04	AE	9F	00243	PUSHAB	OPERAND_NUMBER	1208
				02	DD	00246	PUSHL	#2	
				63	11	00248	BRB	31\$	
		52		05	DD	0024A	MOVL	#5, COUNT	1214
			04	AE	9F	0024D	PUSHAB	OPERAND_NUMBER	
				02	DD	00250	PUSHL	#2	
		66	EC	AD	9F	00252	PUSHAB	ENCODE	
				03	FB	00255	CALLS	#3, PARSE OPERAND	
		F2		52	F4	00258	SQBGEQ	COUNT, 27\$	
				56	11	0025B	BRB	32\$	1213
			04	AE	9F	0025D	PUSHAB	OPERAND_NUMBER	1220
7E	09	A4	04	00	EF	00260	EXTZV	#0, #4, -9(OPCODE_ENTRY), -(SP)	
			EC	AD	9F	00266	PUSHAB	ENCODE	
		66		03	FB	00269	CALLS	#3, PARSE OPERAND	
			04	AE	9F	0026C	PUSHAB	OPERAND_NUMBER	1221
7E	09	A4	04	00	EF	0026F	EXTZV	#0, #4, -9(OPCODE_ENTRY), -(SP)	
				27	11	00275	BRB	30\$	
			04	AE	9F	00277	PUSHAB	OPERAND_NUMBER	1228
7E	09	A4	04	04	EF	0027A	EXTZV	#4, #4, -9(OPCODE_ENTRY), -(SP)	
			EC	AD	9F	00280	PUSHAB	ENCODE	
		66		03	FB	00283	CALLS	#3, PARSE OPERAND	
			04	AE	9F	00286	PUSHAB	OPERAND_NUMBER	1229
7E	09	A4	04	00	EF	00289	EXTZV	#0, #4, -9(OPCODE_ENTRY), -(SP)	

			66	EC	AD	9F	0028F	PUSHAB	ENCODE	:	
					03	FB	00292	CALLS	#3, PARSE_OPERAND	:	
7E	09	A4	04	04	AE	9F	00295	PUSHAB	OPERAND_NUMBER	:	1230
					04	EF	00298	EXTZV	#4, #4, -9(_OPCODE_ENTRY), -(SP)	:	
			66	EC	AD	9F	0029E	PUSHAB	ENCODE	:	
					03	FB	002A1	CALLS	#3, PARSE_OPERAND	:	
7E	09	A4	04	04	AE	9F	002A4	PUSHAB	OPERAND_NUMBER	:	1231
					00	EF	002A7	EXTZV	#0, #4, -9(_OPCODE_ENTRY), -(SP)	:	
			66	EC	AD	9F	002AD	PUSHAB	ENCODE	:	
					03	FB	002B0	CALLS	#3, PARSE_OPERAND	:	
	08		BC	F4	AD	90	002B3	MOVB	ENCODE+8, -@OUTPUT_BUFFER	:	1241
			50		01	00	002B8	MOVL	#1, R0	:	1242
					04	00	002BB	RET		:	1243

; Routine Size: 700 bytes, Routine Base: DBG\$CODE + 0520

```

1129 1244 1 GLOBAL ROUTINE DBG$Opcode_Index(Input_Desc : REF dbg$stg_desc) =
1130 1245 ++
1131 1246
1132 1247 Return the index of DBG$Opcode_Name_Table of the opcode MNemonic
1133 1248 Passed in Input_Desc.
1134 1249
1135 1250 Input_desc is modified to point after the MNemonic. That is the
1136 1251 MNemonic is consumed. Input_desc must be the Mnemonic only if
1137 1252 Delimiter is not passed.
1138 1253
1139 1254 --
1140 1255 BEGIN
1141 1256 LOCAL
1142 1257 Code,
1143 1258 Index;
1144 1259
1145 1260 Index = Opcode_Name_Index ( .Input_desc );
1146 1261 Code = .DBG$OpCode_Name_Table[ .index, 7, 0, 8, 0];
1147 1262 ++
1148 1263 Return the index into the Kind_table
1149 1264 --
1150 1265 RETURN (IF (.DBG$Opcode_Name_Table[ .index, 6, 0, 8, 0] LSS %X'FD')
1151 1266 THEN (.Code)
1152 1267 ELSE ( .Code + %X'100' )
1153 1268 );
1154 1269 END;

```

Get the index into the name

Test for 2 byte opcode!  
1 byte opcode index  
2 byte opcode index

0000V	CF	04	AC	DD	00002	.ENTRY	DBG\$OPCODE_INDEX, Save nothing	1244
	50		01	FB	00005	PUSHL	INPUT_DESC	1260
	51	00000000	0A	C4	0000A	CALLS	#1, OPCODE_NAME_INDEX	
FD	8F	00000000	9A	0000D	91	MULL2	#10, R0	1261
			04	1E	0001E	MOVZBL	DBG\$OPCODE_NAME_TABLE+7[R0], CODE	
	50		51	D0	00020	CMPB	DBG\$OPCODE_NAME_TABLE+6[R0], #253	1265
			04	00	00023	BGEQU	1\$	
	50	0100	C1	9E	00024	MOVL	CODE, R0	1266
			04	00	00029	RET		
						MOVAB	256(R1), R0	1267
						RET		1269

; Routine Size: 42 bytes. Routine Base: DBG\$CODE + 07DC



```

1156 1270 1 ROUTINE Opcode_Name_Index(Input_Desc : REF dbg$stg_desc,Delimiter) =
1157 1271 1
1158 1272 1 FUNCTIONAL DESCRIPTION:
1159 1273 1
1160 1274 1 Opcode_Name_Index returns the index into DBG$OPCODE_NAME_TABLE
1161 1275 1 given a mnemonic.
1162 1276 1
1163 1277 1 FORMAL PARAMETERS:
1164 1278 1
1165 1279 1 Input_Desc - A string descriptor by reference, describing the
1166 1280 1 Mnemonic.
1167 1281 1
1168 1282 1 Delimiter - An optional parameter, describing the character
1169 1283 1 with which to terminate the Mnemonic, passed
1170 1284 1 by value.
1171 1285 1
1172 1286 1 IMPLICIT INPUTS:
1173 1287 1
1174 1288 1 DBG$Opcode_Name_table
1175 1289 1
1176 1290 1 IMPLICIT OUTPUTS:
1177 1291 1
1178 1292 1 Input_desc is update to consume the Mnemonic.
1179 1293 1
1180 1294 1
1181 1295 1 ROUTINE VALUE:
1182 1296 1
1183 1297 1 The index into DBG$Opcode_Name_Table for the Mnemonic passed
1184 1298 1
1185 1299 1
1186 1300 1 SIDE EFFECTS:
1187 1301 1
1188 1302 1 None
1189 1303 1
1190 1304 1 BEGIN
1191 1305 1 BUILTIN ACTUALCOUNT;
1192 1306 1 LOCAL
1193 1307 1 index,
1194 1308 1 length,
1195 1309 1 limit_LSS,
1196 1310 1 limit_GTR;
1197 1311 1
1198 1312 1
1199 1313 1 IF Actualcount() GTR 1
1200 1314 1 THEN
1201 1315 1 length = Scan_Operand(.Input_Desc,.Delimiter)
1202 1316 1 ELSE
1203 1317 1 BEGIN
1204 1318 1
1205 1319 1 LOCAL Temp_ptr;
1206 1320 1
1207 1321 1 Skip_Leading_Blanks(.Input_Desc);
1208 1322 1
1209 1323 1 Temp_ptr = CH$PTR ( .Input_Desc[ dsc$a_pointer ] );
1210 1324 1
1211 1325 1 !++
1212 1326 1 ! Consume the Mnemonic

```

! Current position in the inp

! Assign the start of the str

```
1213      length = 0;
1214      WHILE .length LSS .Input_Desc[dsc$w_length] DO
1215      BEGIN
1216          LOCAL char : BYTE UNSIGNED;
1217          char = CH$RCHAR_A( Temp_ptr );
1218          IF ((.char GEQU %C'A') AND (.char LEQU %C'Z'))
1219          OR ((.char GEQU %C'0') AND (.char LEQU %C'9'))
1220          THEN
1221              length = .length + 1
1222          ELSE
1223              EXITLOOP;
1224          END;
1225      END;
1226      limit_LSS = 0;
1227      limit_GTR = Opcode_Index + 1;
1228      ++
1229      Binary search of DBG$Opcode_Name_Table for the Mnemonic
1230      --
1231      WHILE ((index = (.limit_LSS + .limit_GTR)/2) NEQ .limit_LSS) DO
1232      BEGIN
1233          SELECTONE ch$compare(6,DBG$Opcode_Name_Table[.index,offset],
1234              .length,.Input_Desc[dsc$a_pointer],%C' ') OF
1235          SET
1236          [-1]: limit_LSS = .index;
1237          [+1]: limit_GTR = .index;
1238          [ 0]:
1239              BEGIN
1240                  Input_Desc[dsc$w_length] = .Input_Desc[dsc$w_length] -.length;
1241                  Input_Desc[dsc$a_pointer] = .Input_Desc[dsc$a_pointer]+.length;
1242                  RETURN index
1243              END;
1244      END;
1245      TES;
1246      END;
1247      SIGNAL(dbg$_badopCode,2,.length,.Input_Desc[dsc$a_pointer]);
1248      RETURN 0;
1249      END;
1250
```

03FC 0000 OPCODE_NAME INDEX:					
56	04	AC	D0 00002	WORD Save R2,R3,R4,R5,R6,R7,R8,R9	1270
01		6C	91 00006	MOVL INPUT_DESC, R6	1315
		0F	1B 00009	CMPB (AP),-#1	1313
	08	AC	DD 0000B	BLEQU 1\$	
		56	DD 0000E	PUSHL DELIMITER	1315
0000V	CF	02	FB 00010	PUSHL R6	
59		50	D0 00015	CALLS #2, SCAN OPERAND	
		51	11 00018	MOVL R0, LENGTH	
		56	DD 0001A	BRB 5\$	
0000V	CF	01	FB 0001C	PUSHL R6	1321
51	04	A6	D0 00021	CALLS #1, SKIP LEADING_BLANKS	
		59	D4 00025	MOVL 4(R6), TEMP_PTR	1323
				CLRL LENGTH	1328

59	66	10	00	ED	00027	28:	CMPZV	#0, #16, (R6), LENGTH	1329	
			1D	15	0002C		BLEQ	58	1332	
		50	81	90	0002E		MOVB	(TEMP_PTR)+, CHAR	1333	
	41	8F	50	91	00031		CMPB	CHAR, #65		
			06	1F	00035		BLSSU	38		
	5A	8F	50	91	00037		CMPB	CHAR, #90		
			0A	1B	00038		BLEQU	48		
		30	50	91	0003D	38:	CMPB	CHAR, #48	1334	
			09	1F	00040		BLSSU	58		
		39	50	91	00042		CMPB	CHAR, #57		
			04	1A	00045		BGTRU	58		
			59	D6	00047	48:	INCL	LENGTH	1336	
			DC	11	00049		BRB	28		
			55	D4	0004B	58:	CLRL	LIMIT_LSS	1342	
		58	8F	3C	0004D		MOVZWL	#323, LIMIT_GTR	1343	
	50	55	58	C1	00052	68:	ADDL3	LIMIT_GTR, LIMIT_LSS, R0	1347	
	54	50	02	C7	00056		DIVL3	#2, R0, INDEX		
		55	54	D1	0005A		CMPL	INDEX, LIMIT_LSS		
			3F	13	0005D		BEQL	108		
	50	54	0A	C5	0005F		MULL3	#10, INDEX, R0	1349	
		57	01	D0	00063		MOVL	#1, R7		
59	20	00000000'EF40	06	2D	00066		CMPCS	#6, DBG\$OPCODE_NAME_TABLE[R0], #32, -		
			04	B6	00070			LENGTH, 24(R6)		
			03	1A	00072		BGTRU	78		
		57	01	D9	00074		SBWC	#1, R7		
	FFFFFFF	8F	57	D1	00077	78:	CMPL	R7, #1	1352	
			05	12	0007E		BNEQ	88		
		55	54	D0	00080		MOVL	INDEX, LIMIT_LSS		
			CD	11	00083		BRB	68		
		01	57	D1	00085	88:	CMPL	R7, #1	1353	
			05	12	00088		BNEQ	98		
		58	54	D0	0008A		MOVL	INDEX, LIMIT_GTR		
			C3	11	0008D		BRB	68		
			57	D5	0008F	98:	TSTL	R7	1354	
			BF	12	00091		BNEQ	68		
		66	59	A2	00093		SUBW2	LENGTH, (R6)	1356	
	04	A6	59	C0	00096		ADDL2	LENGTH, 4(R6)	1357	
		50	54	D0	0009A		MOVL	INDEX, R0	1358	
				04	0009D		RET			
			04	A6	DD	0009E	108:	PUSHL	4(R6)	1362
			59	DD	000A1		PUSHL	LENGTH		
			02	DD	000A3		PUSHL	#2		
			8F	DD	000A5		PUSHL	#164360		
	00000000G	00	04	FB	000AB		CALLS	#4, LIB\$SIGNAL		
			50	D4	000B2		CLRL	R0	1363	
				04	000B4		RET		1364	

; Routine Size: 181 bytes, Routine Base: DBG\$CODE + 0806

03FC 00000 FETCH\_INSTRUCTION:

; Routine Size: 83 bytes, Routine Base: DBG\$CODE + 08BB



```

1268 1380 1 ROUTINE Fetch_Operand(Pointer,Context,Print_Flag,Index_Flag) : NOVALUS =
1269 1381 BEGIN
1270 1382 LOCAL
1271 1383 Mode_Specifier,
1272 1384 Register_Field;
1273 1385
1274 1386
1275 1387 SELECTONE .Context OF
1276 1388 SET
1277 1389 [context_p,context_t]:
1278 1390 BEGIN
1279 1391 Fetch_Operand(.Pointer,context_wu,.Print_Flag,.Index_Flag);
1280 1392 IF .Print_Flag THEN DBGSPrint(Format_AD,T,comma);
1281 1393 Fetch_Operand(.Pointer,context_b,.Print_Flag,.Index_Flag);
1282 1394 END;
1283 1395 [context_m,context_v]:
1284 1396 BEGIN
1285 1397 Fetch_Operand(.Pointer,context_l,.Print_Flag,.Index_Flag);
1286 1398 IF .Print_Flag THEN DBGSPrint(Format_AD,1,comma);
1287 1399 Fetch_Operand(.Pointer,context_b,.Print_Flag,.Index_Flag);
1288 1400 IF .context EQL context_v THEN RETURN;
1289 1401 IF .Print_Flag THEN DBGSPrint(Format_AD,1,comma);
1290 1402 Fetch_Operand(.Pointer,context_b,.Print_Flag,.Index_Flag);
1291 1403 END;
1292 1404 [OTHERWISE]:
1293 1405 BEGIN
1294 1406 Fetch_Instruction(.Pointer,context_bu);
1295 1407 Mode_Specifier = .Op_Buffer[0,4,4,0];
1296 1408 Register_Field = .Op_Buffer[0,0,4,0];
1297 1409
1298 1410 IF (.Print_Flag AND .Index_Flag AND (.Mode_Specifier LEQ 5))
1299 1411 THEN SIGNAL(dbg$_addressmode);
1300 1412
1301 1413 CASE .Mode_Specifier FROM 0 TO 15 OF
1302 1414 SET
1303 1415 [0,1,2,3]: ! Literal
1304 1416 IF (.Print_Flag) THEN
1305 1417 BEGIN
1306 1418 DBGSPrint(Format_AD,3,UPLIT BYTE('S^N'));
1307 1419 SELECTONE .Context OF
1308 1420 SET
1309 1421 [context_f,context_d,
1310 1422 context_g,context_h]:
1311 1423 BEGIN
1312 1424 Operand_Value = (.Operand_Value^4) OR ZX'4000';
1313 1425 Print_Operand(context_f);
1314 1426 END;
1315 1427 [OTHERWISE]:
1316 1428 BEGIN
1317 1429 Print_Operand(Context_b);
1318 1430 END;
1319 1431 TES;
1320 1432 END;
1321 1433
1322 1434 [4,5,6,7,8,9]: ! Various Register Modes
1323 1435 BEGIN
1324 1436 IF (.Mode_Specifier<3,1,0> AND (.Register_Field EQL 15))

```

```
THEN
  BEGIN
    IF NOT .Mode_Specifier
    THEN
      BEGIN
        Fetch_Instruction(.Pointer,.Context);
        IF .Print_Flag THEN
          BEGIN
            DBG$Print(Format_AD,3,UPLIT BYTE('I^N'));
            Print_Operand(.Context);
          END;
        END
      ELSE
        BEGIN
          Fetch_Instruction(.Pointer,context_1);
          IF .Print_Flag THEN
            BEGIN
              DBG$Print(Format_AD,2,UPLIT BYTE('@#'));
              Print_Address(.Operand_Value);
            END;
          END
        ELSE
          BEGIN
            IF .Mode_Specifier EQL 4 THEN
              Fetch_Operand(.Pointer,.Context,.Print_Flag,1);
            IF .Print_Flag THEN
              BEGIN
                BIND Punctuation = UPLIT BYTE(
                  0 : [ ( . ) ] : 0 : Mode 4
                  0 : 0 : 0 : 0 : Mode 5
                  0 : ( . ) : 0 : Mode 6
                  ' - ' : ( . ) : 0 : Mode 7
                  0 : ( . ) : + : Mode 8
                  ' a ' : ( . ) : + : Mode 9
                  - (4*4) : BLOCKVECTOR [10,4,BYTE];

                IF .Punctuation[.Mode_Specifier,0,0,8,0] NEQ 0 THEN
                  DBG$Print(Format_AD,1,Punctuation[.Mode_Specifier,0,0,0,0]);
                IF .Punctuation[.Mode_Specifier,1,0,8,0] NEQ 0 THEN
                  DBG$Print(Format_AD,1,Punctuation[.Mode_Specifier,1,0,0,0]);
                DBG$Print(Format_AC,Register_Name[Register_Field]);
                IF .Punctuation[.Mode_Specifier,2,0,8,0] NEQ 0 THEN
                  DBG$Print(Format_AD,1,Punctuation[.Mode_Specifier,2,0,0,0]);
                IF .Punctuation[.Mode_Specifier,3,0,8,0] NEQ 0 THEN
                  DBG$Print(Format_AD,1,Punctuation[.Mode_Specifier,3,0,0,0]);
                END;
              END;
            END;
          END;
        [10,11,12,13,14,15]: ! Displacement Modes
        BEGIN
          LOCAL Offset_Context;

          Offset_Context = (.Mode_Specifier-10)/2;
          Fetch_Instruction(.Pointer,.Offset_Context);
          IF (.Print_Flag) THEN
```

```
1325 1437
1326 1438
1327 1439
1328 1440
1329 1441
1330 1442
1331 1443
1332 1444
1333 1445
1334 1446
1335 1447
1336 1448
1337 1449
1338 1450
1339 1451
1340 1452
1341 1453
1342 1454
1343 1455
1344 1456
1345 1457
1346 1458
1347 1459
1348 1460
1349 1461
1350 1462
1351 1463
1352 1464
1353 1465
1354 1466
1355 1467
1356 1468
1357 1469
1358 1470
1359 1471
1360 1472
1361 1473
1362 1474
1363 1475
1364 1476
1365 1477
1366 1478
1367 1479
1368 1480
1369 1481
1370 1482
1371 1483
1372 1484
1373 1485
1374 1486
1375 1487
1376 1488
1377 1489
1378 1490
1379 1491
1380 1492
1381 1493
```

```

1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515

```

```

BEGIN
BIND Mode_Char = UPLIT BYTE('BWL') : VECTOR[.BYTE];
IF (.Mode_Specifier) THEN
  DBG$Print(Format_AD,1,UPLIT BYTE('@'));
  DBG$Print(Format_AD,T,Mode_Char[.Offset_Context]);
  DBG$Print(Format_AD,1,UPLIT BYTE('^'));
  IF (.Register_Field EQL 15)
  THEN
    Print_Address(.Operand_Value + ..Pointer)
  ELSE
    BEGIN
    Print_Operand(.Offset_Context);
    DBG$Print(Format_AD,1,UPLIT BYTE('('));
    DBG$Print(Format_AC,Register_Name[.Register_Field]);
    DBG$Print(Format_AD,1,UPLIT BYTE(')'));
    END;
END;
END;
TES;
END;
TES;
END;
TES;
END;
END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
23 5E 53 01178 P.AAK: .ASCII \S^#\
23 5E 49 0117B P.AAL: .ASCII \I^#\
23 40 0117E P.AAM: .ASCII \A#\
00 01180 P.AAN: .BYTE 0
5B 01181 .ASCII \[\
5D 01182 .ASCII \]\
00 01183 .BYTE 0, 0, 0, 0, 0, 0
28 01189 .ASCII \[\
29 0118A .ASCII \)\
00 0118B .BYTE 0
2D 0118C .ASCII \-\
28 0118D .ASCII \(\
29 0118E .ASCII \)\
00 0118F .BYTE 0, 0
28 01191 .ASCII \[\
29 01192 .ASCII \)\
2B 01193 .ASCII \+\
40 01194 .ASCII \a\
2B 01195 .ASCII \(\
29 01196 .ASCII \)\
2B 01197 .ASCII \+\
4C 57 42 01198 P.AAO: .ASCII \BWL\
40 0119B P.AAP: .ASCII \a\
5E 0119C P.AAQ: .ASCII \^#\
2B 0119D P.AAR: .ASCII \(\
29 0119E P.AAS: .ASCII \)\

PUNCTUATION= P.AAN-16
MODE_CHAR= P.AAO

```

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

07FC 00000 FETCH\_OPERAND:

5A	FB	AF	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10	1380
59	000000000	EF	9E	00006	MOVAB	FETCH_OPERAND, R10	
58	000000000G	00	9E	0000D	MOVAB	OP_BUFFER, R9	
57	000000000	EF	9E	00014	MOVAB	DBG\$PRINT, R8	
54	08	AC	D0	0001B	MOVAB	FORMAT_AD, R7	
0B		54	D1	0001F	MOVL	CONTEXT, R4	1387
		25	19	00022	CMPL	R4, #11	1389
0C		54	D1	00024	BLSS	28	
		20	14	00027	CMPL	R4, #12	
7E	0C	AC	7D	00029	BGTR	28	
		0A	DD	0002D	MOVQ	PRINT_FLAG, -(SP)	1391
	04	AC	DD	0002F	PUSHL	#10	
6A		04	FB	00032	PUSHL	POINTER	
0A	0C	AC	E9	00035	CALLS	#4, FETCH_OPERAND	
	0B	A7	9F	00039	BLBC	PRINT_FLAG, 18	1392
		01	DD	0003C	PUSHAB	COMMA	
		57	DD	0003E	PUSHL	#1	
68		03	FB	00040	PUSHL	R7	
7E	0C	AC	7D	00043	CALLS	#3, DBG\$PRINT	
		4D	11	00047	MOVQ	PRINT_FLAG, -(SP)	1393
0D		54	D1	00049	BRB	68	
		51	19	0004C	CMPL	R4, #13	1395
0E		54	D1	0004E	BLSS	78	
		4C	14	00051	CMPL	R4, #14	
	10	AC	DD	00053	BGTR	78	
52	0C	AC	DD	00056	PUSHL	INDEX_FLAG	1397
		52	DD	0005A	MOVL	PRINT_FLAG, R2	
		02	DD	0005C	PUSHL	R2	
	04	AC	DD	0005E	PUSHL	#2	
6A		04	FB	00061	PUSHL	POINTER	
0A		52	E9	00064	CALLS	#4, FETCH_OPERAND	
	0B	A7	9F	00067	BLBC	R2, 38	1398
		01	DD	0006A	PUSHAB	COMMA	
		57	DD	0006C	PUSHL	#1	
68		03	FB	0006E	PUSHL	R7	
	10	AC	DD	00071	CALLS	#3, DBG\$PRINT	
		52	DD	00074	PUSHL	INDEX_FLAG	1399
		7E	D4	00076	PUSHL	R2	
	04	AC	DD	00078	CLRL	-(SP)	
6A		04	FB	0007B	PUSHL	POINTER	
0E		54	D1	0007E	CALLS	#4, FETCH_OPERAND	
		01	12	00081	CMPL	R4, #14	1400
		52	E9	00084	BNEQ	48	
0A		A7	9F	00087	RET		
	0B	01	DD	0008A	BLBC	R2, 58	1401
		57	DD	0008C	PUSHAB	COMMA	
68		03	FB	0008E	PUSHL	#1	
	10	AC	DD	00091	PUSHL	R7	
		52	DD	00094	CALLS	#3, DBG\$PRINT	
		7E	D4	00096	PUSHL	INDEX_FLAG	1402
	04	AC	DD	00098	PUSHL	R2	
					CLRL	-(SP)	
					PUSHL	POINTER	



[illegible]

		57	DD	0013D	PUSHL	R7		
	68	03	FB	0013F	CALLS	#3, DBGSPRINT		
		54	DD	00142	PUSHL	R4		1446
0000V	CF	01	FB	00144	CALLS	#1, PRINT_OPERAND		
		04	00	00149	RET			1438
		02	DD	0014A	PUSHL	#2		1451
		55	DD	0014C	PUSHL	R5		
AD	AA	02	FB	0014E	CALLS	#2, FETCH_INSTRUCTION		
	1F	56	E9	00152	BLBC	R6, 168		1452
		76	A7	9F	PUSHAB	P, AAM		1454
		02	DD	00158	PUSHL	#2		
		57	DD	0015A	PUSHL	R7		
	68	03	FB	0015C	CALLS	#3, DBGSPRINT		
		69	DD	0015F	PUSHL	OPERAND_VALUE		1455
		00A5	31	00161	BRW	238		
	04	52	D1	00164	CMPL	MODE_SPECIFIER, #4		1461
		0B	12	00167	BNEQ	168		
		01	DD	00169	PUSHL	#1		1462
		0050	8F	BB	PUSHR	#M<R4,R6>		
		55	DD	0016F	PUSHL	R5		
	6A	04	FB	00171	CALLS	#4, FETCH_OPERAND		
	61	56	E9	00174	BLBC	R6, 218		1463
		68	A742	DF	PUSHAL	PUNCTUATION[MODE_SPECIFIER]		1474
		9E	95	0017B	TSTB	@(SP)+		
		0B	13	0017D	BEQL	178		
		68	A742	DF	PUSHAL	PUNCTUATION[MODE_SPECIFIER]		1475
		01	DD	00183	PUSHL	#1		
		57	DD	00185	PUSHL	R7		
	68	03	FB	00187	CALLS	#3, DBGSPRINT		
		69	A742	DF	PUSHAL	PUNCTUATION+1[MODE_SPECIFIER]		1476
		9E	95	0018E	TSTB	@(SP)+		
		0B	13	00190	BEQL	188		
		69	A742	DF	PUSHAL	PUNCTUATION+1[MODE_SPECIFIER]		1477
		01	DD	00196	PUSHL	#1		
		57	DD	00198	PUSHL	R7		
	68	03	FB	0019A	CALLS	#3, DBGSPRINT		
		80	A743	DF	PUSHAL	REGISTER_NAME[REGISTER_FIELD]		1478
		04	A7	9F	PUSHAB	FORMAT AC		
	68	02	FB	001A4	CALLS	#2, DBGSPRINT		
		6A	A742	DF	PUSHAL	PUNCTUATION+2[MODE_SPECIFIER]		1479
		9E	95	001AB	TSTB	@(SP)+		
		0B	13	001AD	BEQL	198		
		6A	A742	DF	PUSHAL	PUNCTUATION+2[MODE_SPECIFIER]		1480
		01	DD	001B3	PUSHL	#1		
		57	DD	001B5	PUSHL	R7		
	68	03	FB	001B7	CALLS	#3, DBGSPRINT		
		68	A742	DF	PUSHAL	PUNCTUATION+3[MODE_SPECIFIER]		1481
		9E	95	001BE	TSTB	@(SP)+		
		74	13	001C0	BEQL	268		
		68	A742	DF	PUSHAL	PUNCTUATION+3[MODE_SPECIFIER]		1482
		67	11	001C6	BRB	258		
	50	F6	A2	9E	MOVAB	-10(R2), R0		1491
54	50	02	C7	001CC	DIVL3	#2, R0, OFFSET_CONTEXT		
		54	DD	001D0	PUSHL	OFFSET_CONTEXT		1492
		55	DD	001D2	PUSHL	R5		
AD	AA	02	FB	001D4	CALLS	#2, FETCH_INSTRUCTION		
	5B	56	E9	001D8	BLBC	R6, 268		1493

08		52	E9	001DB	BLBC	MODE SPECIFIER, 228	1496
	0093	C7	9F	001DE	PUSHAB	P.AAR	1497
		01	DD	001E2	PUSHL	#1	
		57	DD	001E4	PUSHL	R7	
68		03	FB	001E6	CALLS	#3, DBG\$PRINT	
	0090	C7	9F	001E9	PUSHAB	MODE_CHAR[OFFSET_CONTEXT]	1498
		01	DD	001EE	PUSHL	#1	
		57	DD	001F0	PUSHL	R7	
68		03	FB	001F2	CALLS	#3, DBG\$PRINT	
	0094	C7	9F	001F5	PUSHAB	P.AAR	1499
		01	DD	001F9	PUSHL	#1	
		57	DD	001FB	PUSHL	R7	
68		03	FB	001FD	CALLS	#3, DBG\$PRINT	
0F		53	D1	00200	CMPL	REGISTER_FIELD, #15	1500
		0A	12	00203	BNEQ	248	
7E		65	C1	00205	ADDL3	(R5), OPERAND VALUE, -(SP)	1502
	0000V	CF	01	FB	00209	CALLS	#1, PRINT_ADDRESS
			04	0020E	RET		
		54	DD	0020F	PUSHL	OFFSET_CONTEXT	1505
	0000V	CF	01	FB	00211	CALLS	#1, PRINT_OPERAND
			C7	9F	00216	PUSHAB	P.AAR
			01	DD	0021A	PUSHL	#1
			57	DD	0021C	PUSHL	R7
68		03	FB	0021E	CALLS	#3, DBG\$PRINT	
	B0	A7	DF	00221	PUSHAL	REGISTER_NAME[REGISTER_FIELD]	1507
	04	A7	9F	00225	PUSHAB	FORMAT AC	
68		02	FB	00228	CALLS	#2, DBG\$PRINT	
	0096	C7	9F	0022B	PUSHAB	P.AAS	1508
		01	DD	0022F	PUSHL	#1	
		57	DD	00231	PUSHL	R7	
68		03	FB	00233	CALLS	#3, DBG\$PRINT	
		04	00236	268:	RET		1515

; Routine Size: 567 bytes, Routine Base: DBG\$CODE + 090E

```
1405 1516 1 ROUTINE Parse_Operand(EnCode,Context,Operand_number) : NOVALUE = ! M007
1406 1517 BEGIN
1407 1518 MAP EnCode : REF BLOCK [20,BYTE] FIELD(EnCode_Fields);
1408 1519 LABEL scan,trim,mode,Addr;
1409 1520 LOCAL
1410 1521 Regnum,R Mode, ! Index register and addressing mode
1411 1522 Indexed_Regnum, ! For saving the indexed regi
1412 1523 Displacement_size_needed, ! Displacement sized needed f
1413 1524 Length_Parsed, !-Operand lengths (unparsed and total)
1414 1525 Defer_Address, ! Operand Mode and Address
1415 1526 Buffer : REF VECTOR [,BYTE];
1416 1527
1417 1528 MACRO Report_Error = (SIGNAL(dbg$_opsyntax,1,..Operand_number);RETURN)%; ! saves Code M007
1418 1529
1419 1530 SELECTONE .Context OF
1420 1531 SET
1421 1532 [context_p,context_t]:
1422 1533 BEGIN
1423 1534 Parse_Operand(.EnCode,context_wu,..Operand_number); ! M007
1424 1535 Parse_Operand(.EnCode,context_b,..Operand_number); ! M007
1425 1536 RETURN;
1426 1537 END;
1427 1538 [context_m,context_v]:
1428 1539 BEGIN
1429 1540 Parse_Operand(.EnCode,context_l,..Operand_number); ! M007
1430 1541 Parse_Operand(.EnCode,context_b,..Operand_number); ! M007
1431 1542 IF .context EQL context_m THEN Parse_Operand(.EnCode,context_b,..Operand_number);! M007
1432 1543 RETURN;
1433 1544 END;
1434 1545
1435 1546 [OTHERWISE]:
1436 1547 BEGIN
1437 1548 .Operand_number = ..Operand_number + 1; ! A007
1438 1549 Length = Parsed = Scan_Operand(EnCode[Enc_Input_Desc],%C',');
1439 1550 Buffer = .EnCode[Enc_Input_Buffer];
1440 1551 WHILE (.Length GTR 0) AND (.Buffer[.Length-1] EQL %C' ') DO Length = .Length - 1;
1441 1552 IF .Length LEQ 0 THEN SIGNAL(DBG$_INCOMPOPR, 1, ..Operand_number); ! M007
1442 1553 END;
1443 1554 TES;
1444 1555
1445 1556
```



```
1447 scan: BEGIN
1448
1449 Trim block takes the -(Rn)[Rn] off the operand
1450 starting at the back and trimming toward the front
1451
1452 trim:
1453 BEGIN
1454 R_Mode = 0;
1455 Indexed_Regnum = -1;
1456 Displacement_size_needed = 0;
1457 IF .Buffer[.Length-1] EQL XC'i' THEN
1458 BEGIN
1459   DECR Start FROM .Length-1 TO 1 DO
1460     IF .Buffer[.Start-1] EQL XC'i' THEN
1461       BEGIN
1462         Regnum = Parse_Register(Buffer[.Start],.Length-1-.Start);
1463         IF .Regnum LSS 0
1464           THEN
1465             SIGNAL(DBG$_REGREQ, 1, ..Operand_number);
1466             IF .Regnum GEQ 15
1467               THEN
1468                 SIGNAL(DBG$_PCNOTALL, 1, ..Operand_number);
1469                 IF (Length = .Start - 1) LEQ 0
1470                   THEN
1471                     SIGNAL(DBG$_INCOMPOPR, 1, ..Operand_number);
1472                     Indexed_Regnum = .Regnum;
1473                     Regnum = .Regnum + XX'40i;
1474                     Store_Operand(.EnCode,Regnum,1);
1475                     R_Mode = -1;
1476                     EXITLOOP;
1477                     END;
1478                 END;
1479             IF .Buffer[.Length-1] EQL XC'+' THEN
1480               BEGIN
1481                 R_Mode = XX'80i;
1482                 IF (Length = .Length - 1) LEQ 0
1483                   THEN
1484                     SIGNAL(DBG$_INCOMPOPR, 1, ..Operand_number);
1485                     END;
1486             IF .Buffer[.Length-1] EQL XC')' THEN
1487               BEGIN
1488                 DECR Start FROM .Length-1 TO 1 DO
1489                   IF .Buffer[.Start-1] EQL XC'(' THEN
1490                     BEGIN
1491                       Regnum = Parse_Register(Buffer[.Start],.Length-1-.Start);
1492                       IF (.Regnum EQ 15)
1493                         THEN
1494                           SIGNAL(DBG$_PCNOTALL, 1, ..Operand_number);
1495                           IF .R_Mode GTR 0 THEN
1496                             BEGIN
1497                               IF .Regnum LSS 0
1498                                 THEN
1499                                   SIGNAL(DBG$_REGREQ, 1, ..Operand_number);
1500                                   IF (.Indexed_Regnum EQL .Regnum) AND
1501                                     (.R_Mode EQL XX'80i)
1502                                       THEN
1503
```

1504	1614	7	SIGNAL(DBGS_INDBASEQL, 1, ..Operand_number);	!
1505	1615	7	Length = .Start - 1;	! Consume the (Rn)
1506	1616	7	LEAVE trim;	! Done trimming
1507	1617	6	END;	
1508	1618	6	IF .Regnum LSS 0	! Bad register
1509	1619	6	THEN	
1510	1620	6	SIGNAL(DBGS_REGREQ, 1, ..Operand_number);	
1511	1621	6	IF .Start EQL 1	! "(" is the start?
1512	1622	6	THEN R_Mode = XX'60'	! Yes, Construct register def
1513	1623	7	ELSE IF (.Buffer[0] EQL XC'-' ) AND (.Start EQL 2)	! Test for autodecrement
1514	1624	6	THEN R_Mode = XX'70';	! Construct autodecrement
1515	1625	6	IF .R_Mode GTR 0 THEN	! Do we have something?
1516	1626	7	BEGIN	! Yes
1517	1627	7	IF (.Indexed_Regnum EQL .Regnum) AND	! The indexed and base regist
1518	1628	8	(.R_Mode EQL XX'70')	! same if the register is aut
1519	1629	7	THEN	
1520	1630	7	SIGNAL(DBGS_INDBASEQL, 1, ..Operand_number);	!
1521	1631	7	Regnum = .Regnum + .R_Mode;	! Construct the mode and regi
1522	1632	7	Store_Operand(.EnCode,Regnum,1);	! Store the byte
1523	1633	7	LEAVE scan;	! Done scanning
1524	1634	6	END;	
1525	1635	6	Length = .Start - 1;	! Consume the (Rn)
1526	1636	6	R_Mode = XX'A0'; ! Byte, Word, or Long handled later	! Assume simple displacement
1527	1637	6	Displacement_size_needed = 1;	! We don't know the displacem
1528	1638	6	LEAVE trim;	! Done trimming
1529	1639	5	END;	
1530	1640	5	END	
1531	1641	5	ELSE IF .R_Mode GTR 0 THEN Report_Error;	! Autoincrement and no "'
1532	1642	5	END;	
			! End of block 'trim'	

```

++
We get here with Regnum and R_Mode indicating the current
state of the parse. Possible values are :-

R_Mode      Regnum      Description
0            *          no register operands seen
-1           *          [Rn] has been generated
80           n          (Rn)+ parsed (not generated)
A0           n          (Rn) seen - displacement expected
                        (mode will be changed if needed)

IF (.Length GTR 0) AND (.Buffer[0] EQL %C'@') THEN
    BEGIN
        Length = .Length - 1;
        Buffer = .Buffer + 1;
        Defer = %X'10';
    END
ELSE
    Defer = 0;

IF .R_Mode EQL %X'80' THEN
    BEGIN
        IF .Length NEQ 0 THEN SIGNAL(DBG$ OPSYNTAX, 1, ..Operand_number);
        Regnum = .Regnum + .R_Mode + .Defer;
        Store_Operand(.EnCode, Regnum, 1);
        LEAVE scan;
    END;

++
By now (Rn), (Rn)+, -(Rn), @ (Rn)+ are done

IF (.Length LEQ 0) THEN SIGNAL(DBG$ INCOMPOPR, 1, ..Operand_number);

BEGIN
IF (.Buffer[0] EQL %C'#') THEN
    BEGIN
        Buffer = .Buffer + 1;
        Length = .Length - 1;
        IF (.Length LEQ 0) OR (.R_Mode GTR 0) THEN Report_Error;
        IF (.Defer EQL 0) AND (.R_Mode LSS 0) THEN Report_Error;
        R_Mode = %X'80';
        Regnum = %X'0F';
    END
ELSE
    BEGIN
        IF (.Length LSS 2) OR (.Buffer[1] NEQ %C'^') THEN
            BEGIN
                IF (.R_Mode NEQ 0) OR (.Defer NEQ 0)
                OR (Parse_Register(.Buffer, .Length) LSS 0)
                THEN
                    BEGIN
                        IF .R_Mode LEQ 0
                        THEN
                            BEGIN
                                Regnum = 15;
                                R_Mode = %X'A0';
                            END
                        ELSE
                            BEGIN
                                Regnum = 15;
                                R_Mode = %X'0F';
                            END
                        END
                    END
                END
            END
        END
    END
END

```

1591	1700	8	Displacement_size_needed = 1;	! The displacement size must
1592	1701	7	END;	! We must have an Addr-expr
1593	1702	7	LEAVE Addr;	! Simple register mode
1594	1703	6	END;	! Store it
1595	1704	6	Regnum = Parse_Register(.Buffer, .Length) + XX'50';	! Done
1596	1705	6	Store_Operand(.Encode, Regnum, 1);	
1597	1706	6	LEAVE scan;	
1598	1707	6	END	
1599	1708	5	ELSE	
1600	1709	6	BEGIN	! We have ?^xxxx
1601	1710	6	BIND Mode_Char = UPLIT BYTE('BWLGIS') : VECTOR[6,BYTE];	! The possible ?
1602	1711	6	INCR Index FROM 0 TO 5 DO	! Scan down the list
1603	1712	6	IF (.Buffer[0] EQL .Mode_Char[Index]) THEN	! Did we find one
1604	1713	7	BEGIN	
1605	1714	7	Buffer = .Buffer + 2;	! Consume the ^?
1606	1715	7	Length = .Length - 2;	
1607	1716	8	IF (.Length LEQ 0)	! There must be more
1608	1717	7	THEN	
1609	1718	7	SIGNAL(DBGS_INCOMPOPR, 1, ..Operand_number);	
1610	1719	7	IF .Index LEQ 3 THEN	! One of BWLG
1611	1720	8	BEGIN	
1612	1721	8	IF (.Index EQL 3) AND	! If G^
1613	1722	9	((.R_Mode GTR 0) OR (.Defer NEQ 0))	! must not have (Rn) on end o
1614	1723	8	THEN Report_Error;	
1615	1724	8	IF (.R_Mode LEQ 0) THEN	! When there was no (Rn)
1616	1725	9	BEGIN	
1617	1726	9	R_Mode = XX'A0';	! Do the right thing for rela
1618	1727	9	Regnum = XX'0F';	! and relative deferred
1619	1728	8	END;	
1620	1729	8	R_Mode = .R_Mode + (.Index*XX'20');	! Construct the mode
1621	1730	8	Displacement_size_needed = 0;	! The displacement size has b
1622	1731	8	END	
1623	1732	7	ELSE	
1624	1733	8	BEGIN	
1625	1734	9	IF (.R_Mode GTR 0) OR (.Defer GTR 0)	! (Rn) and @ not allowed
1626	1735	8	OR (.Buffer[0] NEQ XC'#') THEN Report_Error;	! Must have #
1627	1736	8	R_Mode = XX'80' + (.Mode_Char[Index]^16);	! Construct the right mode
1628	1737	8	Regnum = XX'0F';	! The right register
1629	1738	8	Buffer = .Buffer + 1;	! Consume the #
1630	1739	8	Length = .Length - 1;	! Wrong sign
1631	1740	9	IF (.Length LEQ 0)	! There must be a addr-expr l
1632	1741	8	THEN	
1633	1742	8	SIGNAL(DBGS_INCOMPOPR, 1, ..Operand_number);	!
1634	1743	7	END;	
1635	1744	7	LEAVE Mode;	! Done
1636	1745	6	END;	
1637	1746	6	SIGNAL(DBGS_BWLGISMUS, 1, ..Operand_number);	! Wrong char in front of ^
1638	1747	5	END;	
1639	1748	4	END;	



```
1641 1749 4
1642 1750
1643 1751
1644 1752
1645 1753
1646 1754
1647 1755
1648 1756
1649 1757
1650 1758
1651 1759
1652 1760
1653 1761
1654 1762
1655 1763
1656 1764
1657 1765
1658 1766
1659 1767
1660 1768
1661 1769
1662 1770
1663 1771
1664 1772
1665 1773
1666 1774
1667 1775
1668 1776
1669 1777
1670 1778
1671 1779
1672 1780
1673 1781
1674 1782
1675 1783
1676 1784
1677 1785
1678 1786
1679 1787
1680 1788
1681 1789
1682 1790
1683 1791
1684 1792
1685 1793
1686 1794
1687 1795
1688 1796
1689 1797
1690 1798
1691 1799
1692 1800
1693 1801
1694 1802
1695 1803
1696 1804
1697 1805

IF (((.R_Mode + .Regnum + .Defer) AND %X'FF') EQL %X'8F') THEN
  BEGIN
    +
    We get here if there is an immediate-mode operand.
    We now read the value using the EXPRESSION syntax
    of the current language and convert it to the type
    required by this instruction.
    Short literals are also handled here. These can be
    differentiated from other immediate-mode operands
    by examining the high word of R_Mode (set to %C'S'
    for short literals, and 0 or %C'I' otherwise).
    -
    LOCAL
      Val_desc_target : REF DBG$VALDESC,
      Val_desc_source : REF DBG$VALDESC,
      vms_desc_source : BLOCK[ 8, BYTE ],
      vms_desc_target : BLOCK[ 8, BYTE ];
    +
    IF .R_Mode<16,8,0> EQLU %C'S' THEN
      SELECT ONE .Context OF
        SET
          [context_f,context_d,
           context_g,context_h]:
          context = context_f;
        [OTHERWISE]:
          context = context_L;
      TES;
    ++
    Set up the source
    --
    vms_desc_source[dsc$b_class] = dsc$b_class_s;
    vms_desc_source[dsc$b_dtype] = DSC$b_DTYPE_T;
    vms_desc_source[dsc$b_length] = .Length;
    vms_desc_source[dsc$a_pointer] = .Buffer;
    +
    Set up the target
    --
    vms_desc_target[dsc$b_class] = dsc$b_class_s;
    vms_desc_target[dsc$b_dtype] = .Data_Type[.Context];
    vms_desc_target[dsc$b_length] = .Data_Size[.Context];
    vms_desc_target[dsc$a_pointer] = Op_Buffer;
    +
    Op_Buffer[0,0,32,0] = 0;
    Op_Buffer[4,0,32,0] = 0;
    Op_Buffer[8,0,32,0] = 0;
    Op_Buffer[12,0,32,0] = 0;
    +
    Val_desc_source = dbg$make_val_desc( vms_desc_source, dbg$b_v_value_desc );! Convert to a value
    +
    Test for negatives and fix things up
    +
    DBGCONV_TEXT_VALUE does not accept the negative sign
    in the value, but it requires a flag in the value
    descriptor.
```

```

1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754

```

```

--
IF CHSRCHAR( .Val_desc_source[ DBG$L_VALUE_POINTER ] ) EQL %C'- '
THEN
    BEGIN
        Val_desc_source[ DBG$L_VALUE_POINTER ] = .Val_desc_source[ DBG$L_VALUE_POINTER ] + 1;
        Val_desc_source[ DBG$W_VALUE_LENGTH ] = .Val_desc_source[ DBG$W_VALUE_LENGTH ] - 1;
        Val_desc_source[ DBG$W_VALUE_SIGN_CODE ] = TOKEN$K_NEGCONST;
    END;

++
-- Send in the right radix so it will do the right conversion
Val_desc_source[ DBG$W_VALUE_TOKENCODE ] =
    (SELECT ONE .DBG$GB_RADIX[DBG$B_RADIX_INPUT] OF
        SET
            [DBG$K_BINARY] : TOKEN$K_BIN_INTEGER;
            [DBG$K_OCTAL] : TOKEN$K_OCT_INTEGER;
            [DBG$K_HEX] : TOKEN$K_HEX_INTEGER;
            [DBG$K_DECIMAL] : TOKEN$K_INTEGER;
            [OTHERWISE] : $DBG_ERROR( 'DBGENCDEC\PARSER_OPERAND unexpected radix');
    TES);

++
-- Make the target value descriptor
Val_desc_target = dbg$make_val_desc( vms_desc_target, dbg$k_value_desc );
Val_desc_target = DBG$CONV_TEXT_VALUE( .Val_desc_source,          ! Convert the number
                                         .Val_desc_target,
                                         .vms_desc_target[dsc$b_dtype]);

++
-- Move the converted value in
-- normally it is already in Op_buffer but in the case
-- of negative numbers the routines used change the
-- pointer rather than move the data.
CHSMOVE( .Val_desc_target[ DBG$W_VALUE_LENGTH ],
          CHSPTR( .Val_desc_target[ DBG$L_VALUE_POINTER ] ),
          CHSPTR( Op_buffer ) );

IF ( (.R_Mode<16,8,0> EQL %C'S') OR ( (.R_Mode<16,8,0> EQL 0) AND
    (.Op_Buffer[0,0,32,0] LSSU %X'40') AND (.Op_Buffer[4,0,32,0] EQL 0) AND
    ((.Context EQL context_b) OR (.Context EQL context_bu) OR
    (.Context EQL context_w) OR (.Context EQL context_wu) OR
    (.Context EQL context_l) OR (.Context EQL context_q))) THEN
    BEGIN
        BUILTIN ROT;
        IF (.Context EQL context_f) THEN
            Operand_value = ROT( (.Operand_Value XOR %X'4000'), -4);
        IF (.Operand_Value GTRU %X'3F') THEN Report_Error;
        Store_Operand( .Encode, Operand_Value, 1);
    END
ELSE
    BEGIN
        Store_Operand( .Encode, UPLIT BYTE(%X'8F'), 1);
        Store_Operand( .Encode, Operand_Value, .Data_Size[.Context]);
    END

```

DBGENCDEC  
V04-000

8 9  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 85  
(26)

: 1755  
: 1756  
: 1757  
: 1758

1863 5  
1864 5  
1865 4  
1866 3

END;  
LEAVE scan;  
END;  
END;

! End Addr.

```
1760 1867
1761 1868
1762 1869
1763 1870
1764 1871
1765 1872
1766 1873
1767 1874
1768 1875
1769 1876
1770 1877
1771 1878
1772 1879
1773 1880
1774 1881
1775 1882
1776 1883
1777 1884
1778 1885
1779 1886
1780 1887
1781 1888
1782 1889
1783 1890
1784 1891
1785 1892
1786 1893
1787 1894
1788 1895
1789 1896
1790 1897
1791 1898
1792 1899
1793 1900
1794 1901
1795 1902
1796 1903
1797 1904
1798 1905
1799 1906
1800 1907
1801 1908
1802 1909
1803 1910
1804 1911
1805 1912
1806 1913
1807 1914
1808 1915

!+
! Here if we have an register/displacement operand,
! or an absolute address (explicit or implicit).
IF NOT Parse_Expression(-1,.Buffer,.Length,Address) THEN SIGNAL(DBG$INVEXPR,1,..Operand_number);
IF (Check_Register(.Address) GEQ 0) THEN SIGNAL(DBG$INVEXPR,1,..Operand_number);

IF (.R_Mode EQL ZX'100') THEN R_Mode = ! G^<address>
    (IF (.Address GEQU ZX'40000000') THEN ZX'90' ELSE ZX'E0');

IF .Displacement_size_needed THEN
    R_Mode = (IF (.Regnum EQL 15)
    THEN
        (SELECTONE .Address -
            (1 + .Encode[Enc_Final_Address] +
            .Encode[Enc_Output_Length]) OF
            SET
            [-127 TO 127]: ZX'A0';
            [-32766 TO 32769]: ZX'C0';
            [OTHERWISE]: ZX'E0';
            TES)
        ELSE
            (SELECTONE .Address OF
            SET
            [-128 TO 127]: ZX'A0';
            [-32768 TO 32767]: ZX'C0';
            [OTHERWISE]: ZX'E0';
            TES));

    Length = (IF ((.R_Mode EQL ZX'90') OR (.R_Mode EQL ZX'80'))
    THEN 4
    ELSE 1^((.R_Mode-ZX'A0')/ZX'20'));
    IF (.Regnum EQL 15) AND (.R_Mode NEQ ZX'80') AND (.R_Mode NEQ ZX'90')
    THEN Address = .Address - (1 + .Length + .Encode[Enc_Final_Address] +
    .Encode[Enc_Output_Length]);

    R_Mode = .R_Mode + .Defer + .Regnum;
    Store_Operand(.Encode,R_Mode,1);
    Store_Operand(.Encode,Address,.Length);
    END; ! End of block 'scan'

    Encode[Enc_Input_Buffer] = .Encode[Enc_Input_Buffer] + .Parsed;
    Encode[Enc_Input_Length] = .Encode[Enc_Input_Length] - .Parsed;
    IF .Encode[Enc_Input_Length] GTR 0 THEN
        BEGIN
            Encode[Enc_Input_Buffer] = .Encode[Enc_Input_Buffer] + 1;
            Encode[Enc_Input_Length] = .Encode[Enc_Input_Length] - 1;
        END;
    END;

!A005 We need to chose a disp
!A005 PC relative displacemen
!A005 Yes
!A005 Calculate the offset wi
!A005 the operand since we do
!A005
!A005
!A005 Byte Note that the tab
!A005 Word take the lack of
!A005 Long calculation into
!A005
!A005
!A005
!A005 Byte
!A005 Word
!A005 Long
!A005
! Long operand?
! Regnum of 15 means PC relat
! Absolute and immediate
```

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

```
53 52 41 50 5C 43 45 44 43 53 49 47 4C 57 42 0119F P.AAT: .ASCII \BWLGIS\
78 65 6E 75 20 44 4E 41 52 45 50 4F 5F 52 45 011A5 P.AAU: .ASCII \)DBGENCDEC\<92>\PARSER_OPERAND unexpect\
011B4
```



78 69 64 61 74 63 65 70 011C3  
72 20 64 65 011C7  
BF 011CF

P.AAV: .ASCII \ed radix\  
.BYTE -113

MODE\_CHAR=

P.AAT

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

OFFC 00000 PARSE\_OPERAND:

SE		20	C2	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	1516
55		AC	D0	00005	SUBL2	#32, SP	
OB	08	55	D1	00009	MOVL	CONTEXT, R5	1530
		13	19	0000C	CMPL	R5, #11	1532
OC		55	D1	0000E	BLSS	18	
		0E	14	00011	CMPL	R5, #12	
	0C	AC	DD	00013	BGTR	18	
		0A	DD	00016	PUSHL	OPERAND_NUMBER	1534
	04	AC	DD	00018	PUSHL	#10	
E1	AF	03	FB	0001B	PUSHL	ENCODE	
		28	11	0001F	CALLS	#3, PARSE_OPERAND	
OD		55	D1	00021	BRB	28	1535
		30	19	00024	CMPL	R5, #13	1538
OE		55	D1	00026	BLSS	38	
		2B	14	00029	CMPL	R5, #14	
	0C	AC	DD	0002B	BGTR	38	
		02	DD	0002E	PUSHL	OPERAND_NUMBER	1540
	04	AC	DD	00030	PUSHL	#2	
C9	AF	03	FB	00033	PUSHL	ENCODE	
	0C	AC	DD	00037	CALLS	#3, PARSE_OPERAND	
		7E	D4	0003A	PUSHL	OPERAND_NUMBER	1541
	04	AC	DD	0003C	CLRL	-(SP)	
BD	AF	03	FB	0003F	PUSHL	ENCODE	
OD		55	D1	00043	CALLS	#3, PARSE_OPERAND	
		01	13	00046	CMPL	R5, #13	1542
			04	00048	BEQL	28	
	0C	AC	DD	00049	RET		
		7E	D4	0004C	PUSHL	OPERAND_NUMBER	1539
	04	AC	DD	0004E	CLRL	-(SP)	1548
AB	AF	03	FB	00051	PUSHL	ENCODE	1549
			04	00055	CALLS	#3, PARSE_OPERAND	
	0C	BC	D6	00056	RET		
		2C	DD	00059	INCL	OPERAND_NUMBER	1539
52	04	AC	DD	0005B	PUSHL	#44	1548
		52	DD	0005F	MOVL	ENCODE, R2	1549
0000V	CF	02	FB	00061	PUSHL	R2	
		50	D0	00066	CALLS	#2, SCAN_OPERAND	
	04	6E	D0	00069	MOVL	R0, PARSED	
		A2	D0	0006C	MOVL	PARSED, LENGTH	
		56	D5	00070	MOVL	4(R2), BUFFER	1550
		0B	15	00072	TSTL	LENGTH	1551
20	FF	A647	91	00074	BLEQ	58	
		04	12	00079	CMPB	-1(LENGTH)[BUFFER], #32	
		56	D7	0007B	BNEQ	58	
		F1	11	0007D	DECL	LENGTH	
		56	D5	0007F	BRB	48	
					TSTL	LENGTH	1552

			12	14	00081	BGTR	68		
		0C	BC	DD	00083	PUSHL	OPERAND_NUMBER		
			01	DD	00086	PUSHL	#1		
00000000G	00	00028268	8F	DD	00088	PUSHL	#164456		
			03	FB	0008E	CALLS	#3, LIB\$SIGNAL		
	58		08	AE	D4 00095	CLRL	R_MODE		1563
			01	CE	00098	MNEGL	#1, INDEXED_REGNUM		1564
			5A	D4	0009B	CLRL	DISPLACEMENT_SIZE_NEEDED		1565
5D	8F	FF	A647	91	0009D	CMPB	-1(LENGTH)[BUFFER], #93		1566
			03	13	000A3	BEQL	78		
			008E	31	000A5	BRW	148		
	52		56	D0	000A8	MOVL	LENGTH, START		1568
			0080	31	000AB	BRW	128		
5B	8F	FF	A247	91	000AE	CMPB	-1(START)[BUFFER], #91		1569
			78	12	000B4	BNEQ	128		
50	56		52	C3	000B6	SUBL3	START, LENGTH, R0		1571
		FF	A0	9F	000BA	PUSHAB	-1(R0)		
			6247	9F	000BD	PUSHAB	(START)[BUFFER]		
0000V	CF		02	FB	000C0	CALLS	#2, PARSE_REGISTER		
04	AE		50	D0	000C5	MOVL	R0, REGNUM		
53			04	AE	D0 000C9	MOVL	REGNUM, R3		1572
			12	18	000CD	BGEQ	98		
		0C	BC	DD	000CF	PUSHL	OPERAND_NUMBER		1574
			01	DD	000D2	PUSHL	#1		
00000000G	00	00028278	8F	DD	000D4	PUSHL	#164472		
	0F		03	FB	000DA	CALLS	#3, LIB\$SIGNAL		
			53	D1	000E1	CMPL	R3, #15		1575
			12	19	000E4	BLSS	108		
			0C	BC	DD	000E6	PUSHL	OPERAND_NUMBER	1577
			01	DD	000E9	PUSHL	#1		
00000000G	00	00028270	8F	DD	000EB	PUSHL	#164464		
	56		03	FB	000F1	CALLS	#3, LIB\$SIGNAL		
		FF	A2	9E	000F8	MOVAB	-1(R2), LENGTH		1578
			12	14	000FC	BGTR	118		
		0C	BC	DD	000FE	PUSHL	OPERAND_NUMBER		1580
			01	DD	00101	PUSHL	#1		
00000000G	00	00028268	8F	DD	00103	PUSHL	#164456		
	58		03	FB	00109	CALLS	#3, LIB\$SIGNAL		
04	AE	00000040	53	D0	00110	MOVL	R3, INDEXED_REGNUM		1581
			8F	C0	00113	ADDL2	#64, REGNUM		1582
			01	DD	0011B	PUSHL	#1		1583
		08	AE	9F	0011D	PUSHAB	REGNUM		
		04	AC	DD	00120	PUSHL	ENCODE		
0000V	CF		03	FB	00123	CALLS	#3, STORE_OPERAND		
08	AE		01	CE	00128	MNEGL	#1, R_MODE		1584
			08	11	0012C	BRB	148		1570
	02		52	F5	0012E	SOBGTR	START, 138		1569
			03	11	00131	BRB	148		
			FF78	31	00133	BRW	88		
	2B	FF	A647	91	00136	CMPB	-1(LENGTH)[BUFFER], #43		1589
			1A	12	0013B	BNEQ	158		
08	AE	80	8F	9A	0013D	MOVZBL	#128, R_MODE		1591
12			56	F5	00142	SOBGTR	LENGTH, 158		1592
		0C	BC	DD	00145	PUSHL	OPERAND_NUMBER		1594
			01	DD	00148	PUSHL	#1		
00000000G	00	00028268	8F	DD	0014A	PUSHL	#164456		
			03	FB	00150	CALLS	#3, LIB\$SIGNAL		

50	29	FF	A647	91	00157	158:	CMPB	-1(LENGTH)[BUFFER], #41	1597	
			03	13	0015C		BEQL	168		
			00F0	31	0015E		BRW	308		
	52		56	D0	00161	168:	MOVL	LENGTH, START	1599	
			00E2	31	00164	178:	BRW	288		
	28	FF	A247	91	00167	188:	CMPB	-1(START)[BUFFER], #40	1600	
			F6	12	0016C		BNEQ	178		
	56		52	C3	0016E		SUBL3	START, LENGTH, R0	1602	
		FF	A0	9F	00172		PUSHAB	-1(R0)		
			6247	9F	00175		PUSHAB	(START)[BUFFER]		
0000V		CF	02	FB	00178		CALLS	#2, PARSE_REGISTER		
04		AE	50	D0	0017D		MOVL	R0, REGNUM		
	04		53	D0	00181		MOVL	REGNUM, R3	1603	
		OF	53	D1	00185		CMPL	R3, #15		
			12	12	00188		BNEQ	198		
		0C	BC	DD	0018A		PUSHL	OPERAND_NUMBER	1605	
			01	DD	0018D		PUSHL	#1		
000000006	00	00028270	8F	DD	0018F		PUSHL	#164464		
			03	FB	00195		CALLS	#3, LIB\$SIGNAL		
	08		AE	D5	0019C	198:	TSTL	R_MODE	1606	
			3D	15	0019F		BLEQ	228		
			53	D5	001A1		TSTL	R3	1608	
			12	18	001A3		BGEQ	208		
		0C	BC	DD	001A5		PUSHL	OPERAND_NUMBER	1610	
			01	DD	001A8		PUSHL	#1		
		00028278	8F	DD	001AA		PUSHL	#164472		
000000006	00		03	FB	001B0		CALLS	#3, LIB\$SIGNAL		
	53		58	D1	001B7	208:	CMPL	INDEXED_REGNUM, R3	1611	
			1C	12	001BA		BNEQ	218		
00000080	8F	08	AE	D1	001BC		CMPL	R_MODE, #128	1612	
			12	12	001C4		BNEQ	2T8		
		0C	BC	DD	001C6		PUSHL	OPERAND_NUMBER	1614	
			01	DD	001C9		PUSHL	#1		
		00028280	8F	DD	001CB		PUSHL	#164480		
000000006	00		03	FB	001D1		CALLS	#3, LIB\$SIGNAL		
	56		FF	A2	9E	001D8	218:	MOVAB	-1(R2), LENGTH	1615
			78	11	001DC		BRB	318	1616	
			53	D5	001DE	228:	TSTL	R3	1618	
			12	18	001E0		BGEQ	238		
		0C	BC	DD	001E2		PUSHL	OPERAND_NUMBER	1620	
			01	DD	001E5		PUSHL	#1		
		00028278	8F	DD	001E7		PUSHL	#164472		
000000006	00		03	FB	001ED		CALLS	#3, LIB\$SIGNAL		
	01		52	D1	001F4	238:	CMPL	START, #1	1621	
			07	12	001F7		BNEQ	248		
	08	AE	60	8F	9A	001F9	MOVZBL	#96, R_MODE	1622	
			0F	11	001FE		BRB	258		
	2D		67	91	00200	248:	CMPB	(BUFFER), #45	1623	
			0A	12	00203		BNEQ	258		
	02		52	D1	00205		CMPL	START, #2		
			05	12	00208		BNEQ	258		
	08	AE	70	8F	9A	0020A	MOVZBL	#112, R_MODE	1624	
	54	08	AE	D0	0020F	258:	MOVL	R_MODE, R4	1625	
			26	15	00213		BLEQ	278		
	53		58	D1	00215		CMPL	INDEXED_REGNUM, R3	1627	
			1B	12	00218		BNEQ	268		
00000070	8F		54	D1	0021A		CMPL	R4, #112	1628	

			DC	12	12	00221	BNEQ	268			
				BC	DD	00223	PUSHL	30	OPERAND_NUMBER		1630
				01	DD	00226	PUSHL	#1			
				8F	DD	00228	PUSHL	#164480			
000000006	00		00028280	03	FB	0022E	CALLS	#3, LIBSSIGNAL			
	04	AE		54	CO	00235	ADDL2	R4, REGNUM		1631	
				5D	11	00239	BRB	358		1632	
	56			A2	9E	0023B	MOVAB	-1(R2), LENGTH		1635	
08	AE	FF		8F	9A	0023F	MOVZBL	#160, R MODE		1636	
	5A	AO		01	DO	00244	MOVL	#1, DISPLACEMENT_SIZE_NEEDED		1637	
				0D	11	00247	BRB	318		1638	
	02			52	F5	00249	SOBGTR	START, 298		1600	
				08	11	0024C	BRB	318		1597	
				FF16	31	0024E	BRW	188		1600	
				08	AE	D5	00251	TSTL	R MODE	1641	
				65	14	00254	BGTR	388			
				56	D5	00256	TSTL	LENGTH		1655	
				0F	15	00258	BLEQ	328			
40	8F			67	91	0025A	CMPB	(BUFFER), #64			
				09	12	0025E	BNEQ	328			
				56	D7	00260	DECL	LENGTH		1657	
				57	D6	00262	INCL	BUFFER		1658	
	59			10	DO	00264	MOVL	#16, DEFER		1659	
				02	11	00267	BRB	338		1655	
				59	D4	00269	CLRL	DEFER		1662	
	52	08		AE	DO	0026B	MOVL	R MODE, R2		1664	
00000080	8F			52	D1	0026F	CPL	R2, #128			
				22	12	00276	BNEQ	368			
				56	D5	00278	TSTL	LENGTH		1666	
				12	13	0027A	BEQL	348			
			0C	BC	DD	0027C	PUSHL	30	OPERAND_NUMBER		
				01	DD	0027F	PUSHL	#1			
				8F	DD	00281	PUSHL	#164368			
000000006	00		00028210	03	FB	00287	CALLS	#3, LIBSSIGNAL			
04	50			AE	C1	0028E	ADDL3	REGNUM, R2, R0		1667	
AE	52	04		59	C1	00293	ADDL3	DEFER, R0, REGNUM			
	50			7E	11	00298	BRB	468		1668	
				56	D5	0029A	TSTL	LENGTH		1675	
				12	14	0029C	BGTR	378			
			0C	BC	DD	0029E	PUSHL	30	OPERAND_NUMBER		
				01	DD	002A1	PUSHL	#1			
				8F	DD	002A3	PUSHL	#164456			
000000006	00		00028268	03	FB	002A9	CALLS	#3, LIBSSIGNAL			
	23			67	91	002B0	CMPB	(BUFFER), #35		1678	
				21	12	002B3	BNEQ	418			
				57	D6	002B5	INCL	BUFFER		1680	
				56	D7	002B7	DECL	LENGTH		1681	
				03	14	002B9	BGTR	398		1682	
				025A	31	002BB	BRW	698			
	52			D5	002BE	TSTL	R2	388			
	F9			14	002C0	BGTR	388				
	59			D5	002C2	TSTL	DEFER			1683	
	04			12	002C4	BNEQ	408				
	52			D5	002C6	TSTL	R2	388			
	F1			19	002C8	BLSS	388				
08	AE	80		8F	9A	002CA	MOVZBL	#128, R MODE		1684	
04	AE			0F	DO	002CF	MOVL	#15, REGNUM		1685	



	02		00EB	31	002D3	BRW	57\$	1678
			56	D1	002D6	41\$: CMPL	LENGTH, #2	1689
			07	19	002D9	BLSS	42\$	
5E	8F	01	A7	91	002DB	CMPB	1(BUFFER), #94	
			3E	13	002E0	BEQL	47\$	
			52	D5	002E2	42\$: TSTL	R2	1691
			11	12	002E4	BNEQ	43\$	
			59	D5	002E6	TSTL	DEFER	
			0D	12	002E8	BNEQ	43\$	
			56	DD	002EA	PUSHL	LENGTH	1692
			57	DD	002EC	PUSHL	BUFFER	
0000V	CF		02	FB	002EE	CALLS	#2, PARSE_REGISTER	
			50	D5	002F3	TSTL	R0	
			13	18	002F5	BGEQ	45\$	
			52	D5	002F7	43\$: TSTL	R2	1695
			0C	14	002F9	BGTR	44\$	
04	AE		0F	D0	002FB	MOVL	#15, REGNUM	1698
08	AE	A0	8F	9A	002FF	MOVZBL	#160, R MODE	1699
	5A		01	D0	00304	MOVL	#1, DISPLACEMENT_SIZE_NEEDED	1700
			0246	31	00307	44\$: BRW	73\$	1702
			56	DD	0030A	45\$: PUSHL	LENGTH	1704
			57	DD	0030C	PUSHL	BUFFER	
0000V	CF		02	FB	0030E	CALLS	#2, PARSE_REGISTER	
04	AE	50	A0	9E	00313	MOVAB	80(R0), REGNUM	
			01	DD	00318	46\$: PUSHL	#1	1705
		08	AE	9F	0031A	PUSHAB	REGNUM	
			037E	31	0031D	BRW	91\$	
			52	D4	00320	47\$: CLRL	INDEX	1711
00000000	'EF42		67	91	00322	48\$: CMPB	(BUFFER), MODE_CHAR[INDEX]	1712
			7D	12	0032A	BNEQ	55\$	
	57		02	C0	0032C	ADDL2	#2, BUFFER	1714
	56		02	C2	0032F	SUBL2	#2, LENGTH	1715
			12	14	00332	BGTR	49\$	1716
		0C	BC	DD	00334	PUSHL	30PERAND_NUMBER	1718
			01	DD	00337	PUSHL	#1	
			8F	DD	00339	PUSHL	#164456	
00000000G	00	00028268	03	FB	0033F	CALLS	#3, LIB\$SIGNAL	
	03		52	D1	00346	49\$: CMPL	INDEX, #3	1719
			25	14	00349	BGTR	52\$	
			09	12	0034B	BNEQ	50\$	1721
		08	AE	D5	0034D	TSTL	R MODE	1722
			2A	14	00350	BGTR	53\$	
			59	D5	00352	TSTL	DEFER	
			26	12	00354	BNEQ	53\$	
		08	AE	D5	00356	50\$: TSTL	R MODE	1724
			09	14	00359	BGTR	5T\$	
08	AE	A0	8F	9A	0035B	MOVZBL	#160, R MODE	1726
04	AE		0F	D0	00360	MOVL	#15, REGNUM	1727
50	52		05	78	00364	51\$: ASHL	#5, INDEX, R0	1729
	AE		50	C0	00368	ADDL2	R0, R MODE	
			5A	D4	0036C	CLRL	DISPLACEMENT_SIZE_NEEDED	1730
			51	11	0036E	BRB	57\$	1719
		08	AE	D5	00370	52\$: TSTL	R MODE	1734
			07	14	00373	BGTR	53\$	
			59	D5	00375	TSTL	DEFER	
			03	14	00377	BGTR	53\$	
	23		67	91	00379	CMPB	(BUFFER), #35	1735

			03	13	0037C	538:	BEQL	548		
			0197	31	0037E		BRW	698		
50		50	00000000	9A	00381	548:	MOVZBL	MODE_CHAR[INDEX], R0	1736	
	08	50		78	00389		ASHL	#16, R0, R0		
	04	AE	0080	9E	0038D		MOVAB	128(R0), R MODE	1737	
		AE		0F	00393		MOVL	#15, REGNUM	1738	
		25		57	00397		INCL	BUFFER	1739	
				56	00399		SOBGR	LENGTH, 578	1742	
			0C	BC	0039C		PUSHL	OPERAND_NUMBER		
				01	0039F		PUSHL	#1		
			00028268	8F	003A1		PUSHL	#164456		
				11	003A7		BRB	568		
FF73	52	01		05	F1 003A9	558:	ACBL	#5, #1, INDEX 488	1712	
			0C	BC	003AF		PUSHL	OPERAND_NUMBER	1746	
				01	003B2		PUSHL	#1		
			00028288	8F	003B4		PUSHL	#164488		
	50	00000000G	00	03	FB 003BA	568:	CALLS	#3, LIBSSIGNAL		
		08	AE	04	C1 003C1	578:	ADDL3	REGNUM, R_MODE, R0	1749	
			50	59	003C7		ADDL2	DEFER, R0		
		8F	8F	50	91 003CA		CMPB	R0, #143		
				03	13 003CE		BEQL	588		
				017D	31 003D0		BRW	738		
				5B	D4 003D3	588:	CLRL	R11	1767	
	53	8F	0A	AE	91 003D5		CMPB	R_MODE+2, #83		
				16	12 003DA		BNEQ	608		
				5B	D6 003DC		INCL	R11		
			05	55	D1 003DE		CMPL	R5, #5	1770	
			08	0B	19 003E1		BLSS	598		
				55	D1 003E3		CMPL	R5, #8		
				06	14 003E6		BGTR	598		
	08	AC		05	D0 003EB		MOVL	#5, CONTEXT	1772	
				04	11 003EC		BRB	608		
	08	AC		02	D0 003EE	598:	MOVL	#2, CONTEXT	1774	
	1A	AE	010E	8F	B0 003F2	608:	MOVW	#270, VMS_DESC_SOURCE+2	1781	
	18	AE		56	B0 003FB		MOVW	LENGTH, VMS_DESC_SOURCE	1782	
	1C	AE		57	D0 003FC		MOVL	BUFFER, VMS_DESC_SOURCE+4	1783	
	13	AE		01	90 00400		MOVB	#1, VMS_DESC_TARGET+3	1788	
		58		0B	D0 00404		MOVL	CONTEXT, R8	1789	
	12	AE	00000000	48	90 00408		MOVB	DATA_TYPE[R8], VMS_DESC_TARGET+2		
	10	AE	00000000	48	9B 00411		MOVZBW	DATA_SIZE[R8], VMS_DESC_TARGET	1790	
	14	AE	00000000	EF	9E 0041A		MOVAB	OP_BUFFER, VMS_DESC_TARGET+4	1791	
			00000000	EF	7C 00422		CLRQ	OP_BUFFER	1793	
			00000000	EF	7C 00428		CLRQ	OP_BUFFER+8	1795	
		7E		83	9A 0042E		MOVZBL	#131, -(SP)	1798	
			1C	AE	9F 00432		PUSHAB	VMS_DESC_SOURCE		
	00000000G	00		02	FB 00435		CALLS	#2, DBG\$MAKE_VAL_DESC		
		52		50	D0 0043C		MOVL	R0, VAL_DESC_SOURCE		
		2D		B2	91 0043F		CMPB	24(VAL_DESC_SOURCE), #45	1807	
				0B	12 00443		BNEQ	618		
				A2	D6 00445		INCL	24(VAL_DESC_SOURCE)	1810	
				14	B7 00448		DECW	20(VAL_DESC_SOURCE)	1811	
	12	A2		42	9B 0044B		MOVZBW	#66, 18(VAL_DESC_SOURCE)	1812	
		50	00000000G	00	9A 00450	618:	MOVZBL	DBG\$GB_RADIX, R0	1819	
		02		50	91 00457		CMPB	R0, #2	1821	
				05	12 0045A		BNEQ	628		
		50		0A	D0 0045C		MOVL	#10, R0		
				33	11 0045F		BRB	668		

08	50	91	00461	62\$:	CMPB	R0	#8	1822
	05	12	00464		BNEQ	63\$		
50	08	D0	00466		MOVL	#11,	R0	
	29	11	00469		BRB	66\$		
10	50	91	0046B	63\$:	CMPB	R0	#16	1823
	05	12	0046E		BNEQ	64\$		
50	05	D0	00470		MOVL	#5	R0	
	1F	11	00473		BRB	66\$		
0A	50	91	00475	64\$:	CMPB	R0	#10	1824
	05	12	00478		BNEQ	65\$		
50	04	D0	0047A		MOVL	#4,	R0	
	15	11	0047D		BRB	66\$		
	00000000'	EF	9F	0047F	65\$:	PUSHAB	P.AAU	1825
		01	DD	00485		PUSHL	#1	
	00028362	8F	DD	00487		PUSHL	#164706	
00000000G	00	03	FB	0048D		CALLS	#3, LIB\$SIGNAL	
10	A2	50	B0	00494	66\$:	MOVW	R0, 16(VAL_DESC_SOURCE)	1819
	7E	8F	9A	00498		MOVZBL	#122, -(SP)	1831
		14	AE	9F	0049C	PUSHAB	VMS_DESC_TARGET	
00000000G	00	02	FB	0049F		CALLS	#2, DBG\$MAKE_VAL_DESC	
	7E	12	AE	9A	004A6	MOVZBL	VMS_DESC_TARGET+2, -(SP)	1835
			50	DD	004AA	PUSHL	VAL_DESC_TARGET	1834
			52	DD	004AC	PUSHL	VAL_DESC_SOURCE	1833
00000000'	EF	00000000G	00	03	FB	004AE	CALLS	#3, DBG\$CONV_TEXT_VALUE
		18	B0	14	A0	28	004B5	MOVCL
							20(VAL_DESC_TARGET), 224(VAL_DESC_TARGET), -	1845
							OP_BUFFER	
33		5B	E8	004BF	BLBS	R1T,	67\$	1857
	0A	AE	95	004C2	TSTB	R_MODE+2		1847
		68	12	004C5	BNEQ	71\$		
3F	00000000'	EF	D1	004C7	CMPL	OP_BUFFER,	#63	1848
		5F	1A	004CE	BGTRU	71\$		
	00000000'	EF	D5	004D0	TSTL	OP_BUFFER+4		
		57	12	004D6	BNEQ	71\$		
		58	D5	004D8	TSTL	R8		1849
		19	13	004DA	BEQL	67\$		
09		58	D1	004DC	CMPL	R8,	#9	
		14	13	004DF	BEQL	67\$		
01		58	D1	004E1	CMPL	R8,	#1	1850
		0F	13	004E4	BEQL	67\$		
0A		58	D1	004E6	CMPL	R8,	#10	
		0A	13	004E9	BEQL	67\$		
02		58	D1	004EB	CMPL	R8,	#2	1851
		05	13	004EE	BEQL	67\$		
03		58	D1	004F0	CMPL	R8,	#3	
		3A	12	004F3	BNEQ	71\$		
05		58	D1	004F5	CMPL	R8,	#5	1854
		15	12	004F8	BNEQ	68\$		
00000000'	50	00000000'	EF	00004000	XORL3	#16384, OPERAND_VALUE,	R0	1855
	EF		50	FC	ROTL	#-4, R0, OPERAND_VALUE		
		3F	00000000'	EF	D1	0050F	68\$:	1856
			13	1B	00516	CMPL	OPERAND_VALUE,	#63
			0C	BC	DD	00518	69\$:	
			01	DD	0051B	PUSHL	OPERAND_NUMBER	
			8F	DD	0051D	PUSHL	#1	
		00028210	03	FB	00523	PUSHL	#164368	
00000000G	00		04	0052A	CALLS	#3, LIB\$SIGNAL		
			01	DD	0052B	RET		
					70\$:	PUSHL	#1	1857

			18	11	0052D	BRB	72\$		
			01	DD	0052F	PUSHL	#1	1861	
		00000000'	EF	9F	00531	PUSHAB	P.AAV		
		04	AC	DD	00537	PUSHL	ENCODE		
0000V	CF		03	FB	0053A	CALLS	#3, STORE_OPERAND		
	7E	00000000'	EF	9A	0053F	MOVZBL	DATA_SIZE[R8], -(SP)	1862	
		00000000'	EF	9F	00547	PUSHAB	OPERAND_VALUE		
			01	31	0054D	BRW	91\$		
		DC	AE	9F	00550	PUSHAB	ADDRESS	1871	
			56	DD	00553	PUSHL	LENGTH		
			57	DD	00555	PUSHL	BUFFER		
	7E		01	CE	00557	MNEGL	#1, -(SP)		
0000V	CF		04	FB	0055A	CALLS	#4, PARSE_EXPRESSION		
	12		50	EB	0055F	BLBS	R0, 74\$		
		DC	BC	DD	00562	PUSHL	OPERAND_NUMBER		
			01	DD	00565	PUSHL	#1		
00000000G	00	00028290	8F	DD	00567	PUSHL	#164496		
	52	DC	03	FB	0056D	CALLS	#3, LIB\$SIGNAL	1872	
			AE	D0	00574	MOVL	ADDRESS, R2		
0000V	CF		52	DD	00578	PUSHL	R2		
			01	FB	0057A	CALLS	#1, CHECK_REGISTER		
			50	D5	0057F	TSTL	R0		
		DC	12	19	00581	BLSS	75\$		
			BC	DD	00583	PUSHL	OPERAND_NUMBER		
			01	DD	00586	PUSHL	#1		
		00028290	8F	DD	00588	PUSHL	#164496		
00000000G	00		03	FB	0058E	CALLS	#3, LIB\$SIGNAL	1874	
00000100	8F	08	AE	D1	00595	CMPL	R MODE, #256		
			17	12	0059D	BNEQ	78\$		
40000000	8F		52	D1	0059F	CMPL	R2, #1073741824	1875	
			06	1F	005A6	BLSSU	76\$		
	50	90	8F	9A	005A8	MOVZBL	#144, R0		
			04	11	005AC	BRB	77\$		
	50	E0	8F	9A	005AE	MOVZBL	#224, R0		
08	AE		50	D0	005B2	MOVL	R0, R MODE		
	73		5A	E9	005B6	BLBC	DISPLACEMENT_SIZE_NEEDED, 86\$	1877	
	0F	04	AE	D1	005B9	CMPL	REGNUM, #15	1878	
			35	12	005BD	BNEQ	80\$		
	50	04	AC	D0	005BF	MOVL	ENCODE, R0	1881	
50	10	08	A0	C1	005C3	ADDL3	8(R0), 16(R0), R0	1882	
			50	C2	005C9	SUBL2	R0, R2	1881	
			52	D7	005CC	DECL	R2	1880	
FFFFFF81	8F		52	D1	005CE	CMPL	R2, #-127	1884	
			09	19	005D5	BLSS	79\$		
00000080	8F		52	D1	005D7	CMPL	R2, #128		
			26	15	005DE	BLEQ	81\$		
FFFF8002	8F		52	D1	005E0	CMPL	R2, #-32766	1885	
			3B	19	005E7	BLSS	84\$		
00008001	8F		52	D1	005E9	CMPL	R2, #32769		
			2C	15	005F0	BLEQ	83\$		
	50		50	11	005F2	BRB	84\$	1886	
FFFFFF80	8F		52	D1	005F4	CMPL	R2, #-128	1891	
			0F	19	005FB	BLSS	82\$		
0000007F	8F		52	D1	005FD	CMPL	R2, #127		
			06	14	00604	BGTR	82\$		
	50	A0	8F	9A	00606	MOVZBL	#160, R0		
			1C	11	0060A	BRB	85\$		



FFFF8000	8F		52	D1	0060C	82%:	CMPL	R2, #-32768	1892
			0F	19	00613		BLSS	84%	
00007FFF	8F		52	D1	00613		CMPL	R2, #32767	
			06	14	0061C		BGTR	84%	
	50	C0	8F	9A	0061E	83%:	MOVZBL	#192, R0	
			04	11	00622		BRB	85%	
	50	E0	8F	9A	00624	84%:	MOVZBL	#224, R0	1893
08	AE		50	D0	00628	85%:	MOVL	R0, R_MODE	1878
	51	08	AE	D0	0062C	86%:	MOVL	R_MODE, R1	1896
00000090	8F		51	D1	00630		CMPL	RT, #144	
			09	13	00637		BEQL	87%	
00000080	8F		51	D1	00639		CMPL	R1, #128	
			05	12	00640		BNEQ	88%	
	56		04	D0	00642	87%:	MOVL	#4, LENGTH	
			0C	11	00645		BRB	89%	
	50	FF60	C1	9E	00647	88%:	MOVAB	-160(R1), R0	1898
	50		20	C6	0064C		DIVL2	#32, R0	
56	01		50	78	0064F		ASHL	R0, #1, LENGTH	
	0F	04	AE	D1	00653	89%:	CMPL	REGNUM, #15	1899
			29	12	00657		BNEQ	90%	
00000080	8F		51	D1	00659		CMPL	R1, #128	
			20	13	00660		BEQL	90%	
00000090	8F		51	D1	00662		CMPL	R1, #144	
			17	13	00669		BEQL	90%	
	50	04	AC	D0	0066B		MOVL	ENCODE, R0	1900
52	56	10	A0	C1	0066F		ADDL3	16(R0), LENGTH, R2	
	52	08	A0	C0	00674		ADDL2	8(R0), R2	1901
50	0C		52	C3	00678		SUBL3	R2, ADDRESS, R0	1900
	0C		A0	9E	0067D		MOVAB	-1(R0), ADDRESS	
50		FF	59	C1	00682	90%:	ADDL3	DEFER, R1, R0	1903
	08		40	9E	00686		MOVAB	@REGNUM[R0], R_MODE	
		04	D1	DD	0068C		PUSHL	#1	1904
		0C	AE	9F	0068E		PUSHAB	R_MODE	
		04	AC	DD	00691		PUSHL	ENCODE	
0000V	CF		03	FB	00694		CALLS	#3, STORE_OPERAND	
			56	DD	00699		PUSHL	LENGTH	1905
		10	AE	9F	0069B		PUSHAB	ADDRESS	
		04	AC	DD	0069E	91%:	PUSHL	ENCODE	
0000V	CF		03	FB	006A1		CALLS	#3, STORE_OPERAND	
	50	04	AC	D0	006A6		MOVL	ENCODE, R0	1908
	04		6E	C0	006AA		ADDL2	PARSED, 4(R0)	
			6E	A2	006AE		SUBW2	PARSED, (R0)	1909
			05	13	006B1		BEQL	92%	1910
		04	A0	D6	006B3		INCL	4(R0)	1912
			60	B7	006B6		DECW	(R0)	1913
				04	006B8	92%:	RET		1915

; Routine Size: 1721 bytes. Routine Base: DBG\$CODE + 0B45

```
1810 1916 1 ROUTINE Parse_Expression(type,string,length,result) =
1811 1917 2 BEGIN
1812 1918 3
1813 1919 4 Routine Get_Result(Input_Desc,Result,Type) =
1814 1920 5 BEGIN
1815 1921 6
1816 1922 7 Routine Handler(sig_args,mch_args) =
1817 1923 8 BEGIN
1818 1924 9 MAP mch_args : REF BLOCK[.BYTE];
1819 1925 4 EXTERNAL ROUTINE Sys$Unwind : Addressing_Mode(General);
1820 1926 4 mch_args[chf$l_mch_savr0] = 0;
1821 1927 4 Sys$Unwind(0,0);
1822 1928 4 RETURN ss$continue;
1823 1929 3 END; ! End of routine 'handler'
```

			0000	0000	HANDLER:.WORD	Save nothing	
50	08	AC	D0	00002	MOVL	MCH_ARGS, R0	1922
	0C	A0	D4	00006	CLRL	12(R0)	1926
		7E	7C	00009	CLRQ	-(SP)	1927
00000000G	00	02	FB	0000B	CALLS	#2, SYSSUNWIND	1928
	50	01	D0	00012	MOVL	#1, R0	1929
			04	00015	RET		

; Routine Size: 22 bytes. Routine Base: DBG\$CODE + 11FE

```
1824 1930 3 BUILTIN FP;
1825 1931 4 LOCAL valdesc : REF dbg$valdesc;
1826 1932 5
1827 1933 6 .FP = Handler;
1828 1934 7 IF (.Type LSS 0) THEN
1829 1935 8 BEGIN
1830 1936 9 ! Type < 0 means we want an address expression
1831 1937 4
1832 1938 4 IF NOT DBG$Nparse_Address(.Input_Desc,valdesc,
1833 1939 4 dbg$sk_default,tokens$term_none,0) THEN RETURN False;
1834 1940 4
1835 1941 4 IF NOT DBG$Prim_To_Val(.valdesc,dbg$sk_v_value_desc,valdesc)
1836 1942 4 THEN RETURN False;
1837 1943 4
1838 1944 4 .Result = .valdesc[dbg$l_value_pointer];
1839 1945 4 END
1840 1946 3 ELSE
1841 1947 4 BEGIN
1842 1948 4 EXTERNAL dbg$gl_deposit token;
1843 1949 4 LOCAL vms_desc : BLOCK [8,BYTE];
1844 1950 4
1845 1951 4 ! Type > 0 means we want a value expression
1846 1952 4
1847 1953 4 IF NOT DBG$Nparse_Expression(.Input_Desc,dbg$sk_default,
1848 1954 4 valdesc,tokens$term_none,0) THEN RETURN False;
1849 1955 4
1850 1956 4
```

```

1851 1957 4
1852 1958 4
1853 1959 4
1854 1960 4
1855 1961 4
1856 1962 4
1857 1963 4
1858 1964 4
1859 1965 4
1860 1966 4
1861 1967 4
1862 1968 4
1863 1969 4
1864 1970 4
1865 1971 4
1866 1972 2

```

```

vms_desc[dsc$b_class] = dsc$b_class;
vms_desc[dsc$b_dtype] = .Data_Type.Type;
vms_desc[dsc$b_length] = .Data_Size.Type;
vms_desc[dsc$a_pointer] = Op_Buffer;

Op_Buffer[0,0,32,0] = 0; ! Clear high operand bytes
Op_Buffer[4,0,32,0] = 0; ! to extend to quad value
DBG$COVER_DX_DX(.valdesc,
                dbg$make_val_desc(vms_desc,dbg$b_v_value_desc),
                false);
DBG$Eval_Lang_Operator(dbg$gl_deposit_token,.valdesc,
                dbg$make_val_desc(vms_desc,dbg$b_v_value_desc));

END;
RETURN true;
END;

```

```

! Convert the number
! Delete because if follows l

```

! End of routine 'get\_Result'

.EXTRN DBG\$GL\_DEPOSIT\_TOKEN

```

000C 00000 GET_RESULT:
53 00000000' EF 9E 00002 .WORD Save R2,R3
5E OC C2 00009 MOVAB OP_BUFFER, R3
6D DB AF 9E 0000C SUBL2 #12, SP
52 OC AC D0 00010 MOVAB HANDLER, (FP)
31 18 00014 MOVL TYPE, R2
7E 7C 00016 BGEQ 1$
01 DD 00018 CLRQ -(SP)
OC AE 9F 0001A PUSHL #1
04 AC DD 0001D PUSHAB VALDESC
00 05 FB 00020 PUSHL INPUT_DESC
6D 50 E9 00027 CALLS #5, DBG$NPARSE_ADDRESS
7E 83 8F 9A 0002C BLBC R0, 3$
08 08 AE DD 00030 PUSHL SP
00 03 FB 00033 MOVZBL #131, -(SP)
5A 50 E9 0003A PUSHL VALDESC
08 BC 18 A0 D0 00040 CALLS #3, DBG$PRIM_TO_VAL
BC 4C 11 00045 BLBC R0, 3$
08 08 AE 9F 00049 MOVL VALDESC, R0
04 AC DD 0004E MOVL 24(R0), @RESULT
00 05 FB 00051 BRB 2$
07 3C 50 E9 00058 CLRQ -(SP)
06 AE 00000000' EF 42 90 0005F PUSHAB VALDESC
04 AE 00000000' EF 42 9B 00068 PUSHL #1
08 AE 63 9E 00071 PUSHL INPUT_DESC
63 7C 00075 CALLS #5, DBG$NPARSE_EXPRESSION
7E D4 00077 BLBC R0, 3$
7E 83 8F 9A 00079 MOVZBL #131, -(SP)
0C AE 9F 0007D PUSHAB VMS_DESC
00 02 FB 00080 CALLS #2, DBG$MAKE_VAL_DESC

```

```

1919
1934
1935
1939
1942
1945
1935
1954
1957
1958
1959
1960
1962
1964
1965

```

		08	50	DD	00087	PUSHL	R0	
			AE	DD	00089	PUSHL	VALDESC	
00000000G	00		03	FB	0008C	CALLS	#3, DBG\$COVER_DX_DX	1964
	50		01	D0	00093	MOVL	#1, R0	1971
				04	00096	RET		
			50	D4	00097	CLRL	R0	1972
				04	00099	RET		

; Routine Size: 154 bytes, Routine Base: DBG\$CODE + 1214

1867	1973	2	LOCAL
1868	1974		Mark, Status,
1869	1975		Term_Char
1870	1976		: BYTE UNSIGNED,
1871	1977		Term_Addr
1872	1978		: REF VECTOR [1, BYTE],
1873	1979		Local_Desc
1874	1980		: BLOCK [8, BYTE];
1875	1981		Local_Desc[dsc\$b_class] = dsc\$b_class_s;
1876	1982		Local_Desc[dsc\$b_dtype] = dsc\$b_dtype_t;
1877	1983		Local_Desc[dsc\$b_length] = .length;
1878	1984		Local_Desc[dsc\$a_pointer] = .string;
1879	1985		Mark = dbg\$push_tempmem();
1880	1986		Term_Addr = .string + .length;
1881	1987		Term_Char = .Term_Addr[0];
1882	1988		Term_Addr[0] = 13;
1883	1989		Status = Get_Result(Local_Desc, .Result, .Type);
1884	1990		Term_Addr[0] = .Term_Char;
1885	1991		dbg\$pop_tempmem(.Mark);
1886	1992		RETURN .Status;
1887	1993	1	END;

## 003C 00000 PARSE\_EXPRESSION:

					WORD	Save R2,R3,R4,R5			
					SUBL2	#8, SP	1916		
	02	5E	08	C2	00002				
		AE	8F	B0	00005	MOVW	#270, LOCAL_DESC+2	1981	
		6E	AC	B0	0000B	MOVW	LENGTH, LOCAL_DESC	1982	
	04	AE	08	AC	D0	0000F	MOVL	STRING, LOCAL_DESC+4	1983
00000000G	00		00	FB	00014	CALLS	#0, DBG\$PUSH_TEMPMEM	1985	
	55		50	D0	0001B	MOVL	R0, MARK		
52	08	AC	0C	AC	C1	0001E	ADDL3	LENGTH, STRING, TERM_ADDR	1986
		53		62	90	00024	MOVB	(TERM_ADDR), TERM_CHAR	1987
		62		0D	90	00027	MOVB	#13, (TERM_ADDR)	1988
			04	AC	DD	0002A	PUSHL	TYPE	1989
			10	AC	DD	0002D	PUSHL	RESULT	
			08	AE	9F	00030	PUSHAB	LOCAL_DESC	
	FF2E	CF	03	FB	00033	CALLS	#3, GET_RESULT		
		54	50	D0	00038	MOVL	R0, STATUS		
		62	53	90	0003B	MOVB	TERM_CHAR, (TERM_ADDR)	1990	
			55	DD	0003E	PUSHL	MARK	1991	
00000000G	00		01	FB	00040	CALLS	#1, DBG\$POP_TEMPMEM		
	50		54	D0	00047	MOVL	STATUS, R0	1992	



DBGENCDEC  
V04-000

C 10  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 B11gs-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 99  
(28)

04 0004A

RET

; 1993

; Routine Size: 75 bytes, Routine Base: DBG\$CODE + 12AE

```

1889 1994 1 ROUTINE Parse_Register(string,length) =
1890 1995 BEGIN
1891 1996 BIND
1892 1997     Two_char_register = .string + .length - 2 : BLOCKVECTOR [1,WORD],
1893 1998     Three_char_register = .string + .length - 3 : BLOCKVECTOR [1,3];
1894 1999
1895 2000 IF .length LSS 2
1896 2001 THEN
1897 2002     RETURN -1;
1898 2003
1899 2004 SELECTONE .Two_char_register[ 0, 0, 0, 16, 0 ] OF
1900 2005 SET
1901 2006     [ .UPLIT('R0') ]:: RETURN 0;
1902 2007     [ .UPLIT('R1') ]:: RETURN 1;
1903 2008     [ .UPLIT('R2') ]:: RETURN 2;
1904 2009     [ .UPLIT('R3') ]:: RETURN 3;
1905 2010     [ .UPLIT('R4') ]:: RETURN 4;
1906 2011     [ .UPLIT('R5') ]:: RETURN 5;
1907 2012     [ .UPLIT('R6') ]:: RETURN 6;
1908 2013     [ .UPLIT('R7') ]:: RETURN 7;
1909 2014     [ .UPLIT('R8') ]:: RETURN 8;
1910 2015     [ .UPLIT('R9') ]:: RETURN 9;
1911 2016     [ .UPLIT('AP') ]:: RETURN 12;
1912 2017     [ .UPLIT('FP') ]:: RETURN 13;
1913 2018     [ .UPLIT('SP') ]:: RETURN 14;
1914 2019     [ .UPLIT('PC') ]:: RETURN 15;
1915 2020 [ OTHERWISE ]:
1916 2021 BEGIN
1917 2022     IF .length LSS 3
1918 2023     THEN
1919 2024         RETURN -1;
1920 2025
1921 2026     SELECTONE .Three_char_register[ 0, 0, 0, 24, 0 ] OF
1922 2027     SET
1923 2028     [ .UPLIT('R10') ]:: RETURN 10;
1924 2029     [ .UPLIT('R11') ]:: RETURN 11;
1925 2030     [ .UPLIT('R12') ]:: RETURN 12;
1926 2031     [ .UPLIT('R13') ]:: RETURN 13;
1927 2032     [ .UPLIT('R14') ]:: RETURN 14;
1928 2033     [ .UPLIT('R15') ]:: RETURN 15;
1929 2034     [ OTHERWISE ]:
1930 2035         RETURN -1;
1931 2036     TES;
1932 2037 END;
1933 2038 TES;
END;

```

The last 2 characters  
The last 3 characters  
There must be at least 2 ch

First character

Might be a 3 character regi  
There must be at least 3 ch

.PSECT DBG\$PLIT,NOWRT, SHR, PIC,0

00	00	30	52	011D0	P.AAW:	.ASCII	\R0\<0><0>
00	00	31	52	011D4	P.AAX:	.ASCII	\R1\<0><0>
00	00	32	52	011D8	P.AAY:	.ASCII	\R2\<0><0>
00	00	33	52	011DC	P.AAZ:	.ASCII	\R3\<0><0>
00	00	34	52	011E0	P.ABA:	.ASCII	\R4\<0><0>
00	00	35	52	011E4	P.ABB:	.ASCII	\R5\<0><0>
00	00	36	52	011E8	P.ABC:	.ASCII	\R6\<0><0>

00	00	37	52	011EC	P.ABD:	.ASCII	\R7\<0><0>
00	00	38	52	011F0	P.ABE:	.ASCII	\R8\<0><0>
00	00	39	52	011F4	P.ABF:	.ASCII	\R9\<0><0>
00	00	30	41	011FB	P.ABG:	.ASCII	\AP\<0><0>
00	00	50	46	011FC	P.ABH:	.ASCII	\FP\<0><0>
00	00	50	36	01200	P.ABI:	.ASCII	\SP\<0><0>
00	00	43	30	01204	P.ABJ:	.ASCII	\PC\<0><0>
00	00	31	32	01208	P.ABK:	.ASCII	\R10\<0>
00	33	31	32	0120C	P.ABL:	.ASCII	\R11\<0>
00	33	31	32	01210	P.ABM:	.ASCII	\R12\<0>
00	33	31	32	01214	P.ABN:	.ASCII	\R13\<0>
00	33	31	32	01218	P.ABO:	.ASCII	\R14\<0>
00	33	31	32	0121C	P.ABP:	.ASCII	\R15\<0>

0004 00000 PARSE\_REGISTER:

[illegible]

1994  
1997  
2000  
2004  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013

20	A2	50	04	00069	9\$:	RET			2014
		04	D1	0006A		CMPL	R0, P.ABE		
	50	08	12	0006E		BNEQ	10\$, R0		
			D0	00070		MOVL	#8, R0		
		04	04	00073		RET			
24	A2	50	D1	00074	10\$:	CMPL	R0, P.ABF		2015
		04	12	00078		BNEQ	11\$, R0		
	50	09	D0	0007A		MOVL	#9, R0		
		04	04	0007D		RET			
28	A2	50	D1	0007E	11\$:	CMPL	R0, P.ABG		2016
		38	13	00082		BEQL	14\$, R0		
2C	A2	50	D1	00084		CMPL	R0, P.ABH		2017
		3C	13	00088		BEQL	16\$, R0		
30	A2	50	D1	0008A		CMPL	R0, P.ABI		2018
		40	13	0008E		BEQL	18\$, R0		
34	A2	50	D1	00090		CMPL	R0, P.ABJ		2019
		44	13	00094		BEQL	20\$, R0		
	03	08	AC	D1	00096	CMPL	LENGTH, #3		2022
		42	19	0009A		BLSS	21\$, R0		
50	FD	A1	00	EF	0009C	EXTZV	#0, #24, -3(R1), R0		2026
		38	A2	50	D1	000A2	CMPL	R0, P.ABK	2028
			04	12	000A6	BNEQ	12\$, R0		
	50	0A	D0	000AB		MOVL	#10, R0		
		04	04	000AB		RET			
3C	A2	50	D1	000AC	12\$:	CMPL	R0, P.ABL		2029
		04	12	000B0		BNEQ	13\$, R0		
	50	0B	D0	000B2		MOVL	#11, R0		
		04	04	000B5		RET			
40	A2	50	D1	000B6	13\$:	CMPL	R0, P.ABM		2030
		04	12	000BA		BNEQ	15\$, R0		
	50	0C	D0	000BC	14\$:	MOVL	#12, R0		
		04	04	000BF		RET			
44	A2	50	D1	000C0	15\$:	CMPL	R0, P.ABN		2031
		04	12	000C4		BNEQ	17\$, R0		
	50	0D	D0	000C6	16\$:	MOVL	#13, R0		
		04	04	000C9		RET			
48	A2	50	D1	000CA	17\$:	CMPL	R0, P.ABO		2032
		04	12	000CE		BNEQ	19\$, R0		
	50	0E	D0	000D0	18\$:	MOVL	#14, R0		
		04	04	000D3		RET			
4C	A2	50	D1	000D4	19\$:	CMPL	R0, P.ABP		2033
		04	12	000D8		BNEQ	21\$, R0		
	50	0F	D0	000DA	20\$:	MOVL	#15, R0		
		04	04	000DD		RET			
	50	01	CE	000DE	21\$:	MNEGL	#1, R0		2034
		04	04	000E1		RET			2038

; Routine Size: 226 bytes, Routine Base: DBG\$CODE + 12F9

.. 1934	2039	1	
.. 1935	2040	1	
.. 1936	2041	1	
.. 1937	2042	1	
.. 1938	2043	1	
.. 1939	2044	1	ROUTINE Check_Register(Address : UNSIGNED) =
.. 1940	2045	2	BEGIN



```

: 1941      2046  2  EXTERNAL dbg$reg_values;
: 1942      2047      Address = .Address - dbg$reg_values; ! ** NOTE NO "."
: 1943      2048      IF (.Address GTRU 15*ZUPVAL) THEN RETURN -1;
: 1944      2049      IF ((.Address AND (ZUPVAL-1)) NEQ 0) THEN RETURN -1;
: 1945      2050      RETURN .Address/ZUPVAL;
: 1946      2051  1  END;

```

```

                                .EXTRN  DBG$REG_VALUES
                                0000 00000 CHECK_REGISTER:
                                .WORD  Save nothing
                                MOVAB  DBG$REG_VALUES, R0
                                SUBL2  R0, ADDRESS
                                CMPL  ADDRESS, #60
                                BGTRU  1$
                                BITB  ADDRESS, #3
                                BEQL  2$
                                MNEGL  #1, R0
                                RET
                                50      04  AC      04      C7 0001D 2$: DIVL3  #4, ADDRESS, R0
                                04 00022 RET

```

```

: 2044
: 2047
: 2048
: 2049
:
: 2050
: 2051

```

; Routine Size: 35 bytes, Routine Base: DBG\$CODE + 130B

```

: 1948      2052 1 ROUTINE Store_Operand(EnCode,Operand,Length) : NOVALUE =
: 1949      2053      BEGIN
: 1950      2054      MAP
: 1951      2055          EnCode : REF BLOCK [20,BYTE] FIELD(EnCode_Fields),
: 1952      2056          Operand : REF VECTOR [ ,BYTE];
: 1953      2057
: 1954      2058      ch$move(.Length, Operand[0],
: 1955      2059          .EnCode[Enc_Output_Length] + .EnCode[Enc_Output_Buffer]);
: 1956      2060
: 1957      2061      EnCode[Enc_Output_Length] = .EnCode[Enc_Output_Length] + .Length;
: 1958      2062      END;

```

```

                                007C 00000 STORE_OPERAND:
                                .WORD      Save R2,R3,R4,R5,R6
                                MOVL      ENCODE, R6
50      08      56      04      AC      D0 00002      ADDL3      12(R6), 8(R6), R0
60      08      A6      0C      A6      C1 00006      MOVCL3     LENGTH, @OPERAND, (R0)
      08      BC      0C      AC      28 0000C      ADDL2     LENGTH, 8(R6)
      08      A6      0C      AC      C0 00012      RET
                                04 00017

```

```

: 2052
: 2059
:
: 2061
: 2062

```

: Routine Size: 24 bytes.      Routine Base: DBG\$CODE + 13FE

0004 00000 PRINT_OPERAND:			
	5E	0C	C2 00002
00000000G	00	00	FB 00005
	52	50	D0 0000C
03	AE	01	90 0000F
	50	00000000'	EF 9E 00013
02	AE	04 BC40	90 0001A
	50	00000000'	EF 9E 00020
	6E	04 BC40	9B 00027
04	AE	00000000'	EF 9E 0002C
		7E	7C 00034

Save R2
WORD
SUBL2 #12, SP
CALLS #0, DBGSPUSH_TEMPMEM
MOVL R0, MARK
MOVB #1, VMS_DESC+3
MOVAB DATA_TYPE, R0
MOVB @CONTEXT[R0], VMS_DESC+2
MOVAB DATA_SIZE, R0
MOVZBW @CONTEXT[R0], VMS_DESC
MOVAB OP_BUFFER, VMS_DESC+4
CLRR -(SP)

DBGENCDEC  
V04-000

J 10  
16-Sep-1984 00:24:49  
14-Sep-1984 12:16:51

VAX-11 B11ss-32 V4.0-742  
[DEBUG.SRC]DBGENCDEC.B32;1

Page 106  
(31)

	7E	7A	01	DD	00036	PUSHL	#1	
		10	8F	9A	00038	MOVZBL	#122, -(SP)	
			AE	9F	0003C	PUSHAB	VMS_DESC	
00000000G	00		02	FB	0003F	CALLS	#2, DBG\$MAKE_VAL_DESC	
			50	DD	00046	PUSHL	R0	
00000000G	00		04	FB	00048	CALLS	#4, DBG\$PRINT_VALUE	
			52	DD	0004F	PUSHL	MARK	
00000000G	00		01	FB	00051	CALLS	#1, DBG\$POP_TEMPMEM	
			04	00058	RET			

2083  
2084

; Routine Size: 89 bytes,      Routine Base: DBG\$CODE + 1436



```

1983 2085 1 ROUTINE Scan_Operand(Input_Desc : REF dbg$stg_desc,Delimiter) =
1984 2086 BEGIN
1985 2087 BUILTIN ACTUALCOUNT,ACTUALPARAMETER;
1986 2088 LOCAL
1987 2089     Depth,
1988 2090     Local_Desc : dbg$stg_desc,
1989 2091     char : BYTE UNSIGNED;
1990 2092
1991 2093 Depth = (IF Actualcount() LSS 3 THEN 0 ELSE Actualparameter(3)+1);
1992 2094
1993 2095 Skip_Leading_Blanks(.Input_Desc);
1994 2096
1995 2097 ch$move(8,Input_Desc[0,0,0,0],Local_Desc[0,0,0,0]);
1996 2098
1997 2099 WHILE .Local_Desc[dsc$w_length] GEQ 0 DO
1998 2100 BEGIN
1999 2101     LOCAL Target,Length;
2000 2102     char = (IF .Local_Desc[dsc$w_length] EQL 0 THEN 0
2001 2103             ELSE .(.Local_Desc[dsc$a_pointer])<0,8,0>);
2002 2104     Target = 0;
2003 2105     IF .char EQL XX'09' THEN (.Local_Desc[dsc$a_pointer])<0,8,0> = char = XC' ';
2004 2106     IF .char EQL .Delimiter THEN
2005 2107         RETURN (.Local_Desc[dsc$a_pointer] - .Input_Desc[dsc$a_pointer])
2006 2108     ELSE IF .char EQL 0 THEN EXITLOOP;
2007 2109     IF .Delimiter NEQ XC'' THEN
2008 2110         BEGIN
2009 2111             IF (.char GEQ XC'a') AND (.char LEQ XC'z')
2010 2112                 THEN (.Local_Desc[dsc$a_pointer])<0,8,0> = .char - (XC'a'-XC'A')
2011 2113             ELSE IF .char EQL XC'' THEN Target = XC'';
2012 2114             ELSE IF .char EQL XC'(' THEN Target = XC'(';
2013 2115             ELSE IF .char EQL XC'[' THEN Target = XC'[';
2014 2116         END;
2015 2117     Local_Desc[dsc$w_length] = .Local_Desc[dsc$w_length] - 1;
2016 2118     Local_Desc[dsc$a_pointer] = .Local_Desc[dsc$a_pointer] + 1;
2017 2119     IF .Target NEQ 0 THEN
2018 2120         BEGIN
2019 2121             Length = Scan_Operand(Local_Desc,.Target,.Depth) + 1;
2020 2122             Local_Desc[dsc$w_length] = .Local_Desc[dsc$w_length] -.Length;
2021 2123             Local_Desc[dsc$a_pointer] = .Local_Desc[dsc$a_pointer]+.Length;
2022 2124         END;
2023 2125     END;
2024 2126 INFO#252 L1:2099
2025 2127 Test expression is always true
2026 2128 IF .Depth NEQ 0 THEN SIGNAL(dbg$_nodelimtr);
2027 2129 RETURN .Input_Desc[dsc$w_length];
2028 2130 END;

```

	00FC 00000	SCAN_OPERAND:		
5E	0C C2 00002	WORD	Save R2,R3,R4,R5,R6,R7	
03	6C 91 00005	SUBL2	#12, SP	
	04 1E 00008	CMQB	(AP), #3	
	57 D4 0000A	BGEQU	18	
		CLRL	DEPTH	

2085  
2093

57	0C	AC	04	05	11	0000C	BRB	2\$			
		56		01	C1	0000E	ADDL3	#1, 12(AP), DEPTH			2095
				AC	D0	00013	MOVL	INPUT_DESC, R6			
				56	DD	00017	PUSHL	R6			
6E	0000V	CF		01	FB	00019	CALLS	#1, SKIP_LEADING_BLANKS			2097
		66		08	28	0001E	MOVC3	#8, (R6)-LOCAL_DESC			2102
				6E	B5	00022	TSTW	LOCAL_DESC			
				04	12	00024	BNEQ	4\$			
				50	D4	00026	CLRL	R0			
				04	11	00028	BRB	5\$			
		50	04	BE	9A	0002A	MOVZBL	@LOCAL_DESC+4, R0			2103
		52		50	90	0002E	MOVB	R0, CHAR			2102
				51	D4	00031	CLRL	TARGET			2104
		09		52	91	00033	CMPB	CHAR, #9			2105
				07	12	00036	BNEQ	6\$			
		52		20	90	00038	MOVB	#32, CHAR			
08	AC	04		20	90	0003B	MOVB	#32, @LOCAL_DESC+4			
		08		00	ED	0003F	CMPZV	#0, #8, CHAR, DELIMITER			2106
				07	12	00045	BNEQ	7\$			
50	04	AE	04	A6	C3	00047	SUBL3	4(R6), LOCAL_DESC+4, R0			2107
					04	0004D	RET				
				52	95	0004E	TSTB	CHAR			2108
				58	13	00050	BEQL	12\$			
		22	08	AC	D1	00052	CMPL	DELIMITER, #34			2109
				31	13	00056	BEQL	11\$			
	61	8F		52	91	00058	CMPB	CHAR, #97			2111
				0D	1F	0005C	BLSSU	8\$			
	7A	8F		52	91	0005E	CMPB	CHAR, #122			
				07	1A	00062	BGTRU	8\$			
04	BE	52		20	83	00064	SUBB3	#32, CHAR, @LOCAL_DESC+4			2112
				1E	11	00069	BRB	11\$			
		22		52	91	0006B	CMPB	CHAR, #34			2113
				05	12	0006E	BNEQ	9\$			
		51		22	D0	00070	MOVL	#34, TARGET			
				14	11	00073	BRB	11\$			
		28		52	91	00075	CMPB	CHAR, #40			2114
				05	12	00078	BNEQ	10\$			
		51		29	D0	0007A	MOVL	#41, TARGET			
				0A	11	0007D	BRB	11\$			
	5B	8F		52	91	0007F	CMPB	CHAR, #91			2115
				04	12	00083	BNEQ	11\$			
		51	5D	8F	9A	00085	MOVZBL	#93, TARGET			
				6E	B7	00089	DECW	LOCAL_DESC			2117
			04	AE	D6	0008B	INCL	LOCAL_DESC+4			2118
				51	D5	0008E	TSTL	TARGET			2119
				90	13	00090	BEQL	3\$			
			0082	8F	BB	00092	PUSHR	#*M<R1, R7>			2121
			08	AE	9F	00096	PUSHAB	LOCAL_DESC			
	FF62	CF		03	FB	00099	CALLS	#3, SCAN_OPERAND			
				50	D6	0009E	INCL	LENGTH			
		6E		50	A2	000A0	SUBW2	LENGTH, LOCAL_DESC			2122
	04	AE		50	C0	000A3	ADDL2	LENGTH, LOCAL_DESC+4			2123
			FF78	31	000A7	BRW	3\$				2099
				57	D5	000AA	TSTL	DEPTH			2126
				0D	13	000AC	BEQL	13\$			
			00028218	8F	DD	000AE	PUSHL	#164376			
00000000G	00			01	FB	000B4	CALLS	#1, LIB\$SIGNAL			

DBGENCDEC  
V04-000

M 10  
16-Sep-1984 00:24:49 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:16:51 [DEBUG.SRC]DBGENCDEC.B32;1

Page 109  
(32)

50

66 3C 000BB 13\$: MOVZWL (R6), R0  
04 000BE RET

: 2127  
: 2128

; Routine Size: 191 bytes, Routine Base: DBG\$CODE + 148F

```
: 2028      2129 1 ROUTINE Skip_Leading_Blanks(Input_Desc : REF dbg$stg_desc) : NOVALUE =
: 2029      2130 BEGIN
: 2030      2131 WHILE .Input_Desc[dsc$w_length] GTR 0 DO
: 2031      2132 BEGIN
: 2032      2133 LOCAL char : BYTE UNSIGNED;
: 2033      2134 char = .(.Input_Desc[dsc$a_pointer])<0,8,0>;
: 2034      2135 IF (.char NEQ '20') AND (.char NEQ '09') THEN EXITLOOP;
: 2035      2136 Input_Desc[dsc$w_length] = .Input_Desc[dsc$w_length] - 1;
: 2036      2137 Input_Desc[dsc$a_pointer] = .Input_Desc[dsc$a_pointer] + 1;
: 2037      2138 END;
: 2038      2139 END;
```

```
0000 0000 SKIP_LEADING_BLANKS:
50      04 AC D0 00002 .WORD Save nothing
      04 BC B5 00006 1$: MOVL INPUT_DESC, R0
      16 13 00009 TSTW @INPUT_DESC
51      04 B0 90 0000B BEQL 3$
20      51 91 0000F MOVB @4(R0), CHAR
      05 13 00012 CMPB CHAR, #32
      51 91 00014 BEQL 2$
09      08 12 00017 CMPB CHAR, #9
      04 BC B7 00019 BNEQ 3$
      04 A0 D6 0001C 2$: DECW @INPUT_DESC
      E5 11 0001F INCL 4(R0)
      04 00021 3$: BRB 1$
      RET
```

```
: 2129
: 2134
: 2131
: 2134
: 2135
: 2136
: 2137
: 2131
: 2139
```

: Routine Size: 34 bytes, Routine Base: DBG\$CODE + 154E

```
: 2039      2140 1
: 2040      2141 1 END
: 2041      2142 0 ELUDOM
```

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
DBG\$PLIT	4640	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$OWN	16	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON, PIC,ALIGN(2)
DBG\$CODE	5488	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

Library Statistics

----- Symbols ----- Pages Processing



File	Total	Loaded	Percent	Mapped	Time
;\$255SDUA28:[SYSLIB]LIB.L32;1	18619	22	0	1000	00:01.9
;\$255SDUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
;\$255SDUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	56	3	97	00:01.9
;\$255SDUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	0	0	31	00:00.3
;\$255SDUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	15	3	22	00:00.3

; Information: 1  
; Warnings: 0  
; Errors: 0

# COMMAND QUALIFIERS

; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGENCDEC/OBJ=OBJ\$:DBGENCDEC MSRC\$:DBGENCDEC/UPDATE=(ENH\$:DBGENCDEC)

; Size: 5488 code + 4656 data bytes  
; Run Time: 02:23.4  
; Elapsed Time: 07:22.8  
; Lines/CPU Min: 896  
; Lexemes/CPU-Min: 43556  
; Memory Used: 668 pages  
; Compilation Complete



0080 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

